NT132 多用户型多功能加密锁
用户手册



广州飞盾电子有限公司

2009 年 12 月

广州飞盾电子有限公司尽最大努力使本手册的内容完善且正确，对于由本手册导致的任何形式的直接或间接的损失不负有责任。本手册的内容会跟随产品的升级而有所变化。

软件开发协议

广州飞盾电子有限公司（以下简称飞盾电子）的所有产品，包括但不限于：开发工具包，磁盘，光盘，硬件设备和文档，以及未来的所有定单都受本协议的制约。如果您不愿接受这些条款，请在收到后的 7 天内将开发工具包寄回飞盾电子，预付邮资和保险。我们会把货款退还给您，但要扣除运费和适当的手续费。

1. 许可使用

您可以将本软件合并、连接到您的计算机程序中，但其目的只是保护该程序。您可以以存档为目的复制合理数量的拷贝。

2. 禁止使用

除在条款 1 中特别允许的之外，不得复制、反向工程、反汇编、反编译、修改、增加、改进软件、硬件和产品的其它部分。禁止对软件和产品的任何部分进行反向工程，或企图推导软件的源代码。禁止使用产品中的磁性或光学介质来传递、存储非本产品的原始程序或由飞盾电子提供的产品升级的任何数据。禁止将软件放在服务器上传播。

3. 有限担保

飞盾电子保证在自产品交给您之日起的 12 个月内，在正常的使用情况下，硬件和软件存储介质没有重大的工艺和材料上的缺陷。

4. 修理限度

当根据本协议提出索赔时，飞盾电子唯一的责任就是根据飞盾电子的选择，免费进行替换或维修。飞盾电子对更换后的任何产品部件都享有所有权。

保修索赔单必须在担保期内写好，在发生故障 14 天内连同令人信服的证据交给飞盾电子。当将产品返还给飞盾电子或飞盾电子的授权代理商时，须预付运费和保险。

除了在本协议中保证的担保之外，飞盾电子不再提供特别的或隐含的担保，也不再对本协议中所描述的产品负责，包括它们的质量，性能和对某一特定目的的适应性。

5. 责任限度

不管因为什么原因，不管是因合同中的规定还是由于刑事的原因，包括疏忽的原因，而使您及任何一方受到了损失，由我方产品所造成的损失或该产品是起诉的原因或与起诉有间接关系，飞盾电子对您及任何一方所承担的全部责任不超出您购买该产品所支付的货款。在任何情况下，飞盾电子对于由于您不履行责任所导致的损失，或对于数据、利润、储蓄或其它的后续的和偶然的损失，即使飞盾电子被建议有这种损失的可能性，或您根据第 3 方的索赔而提出的任何索赔均不负责任。

6. 协议终止

当您不能遵守本协议所规定的条款时，将终止您的许可和本协议。但条款 2，3，4，5 将继续有效。

广州飞盾电子有限公司
地址：广州市增城新塘.新塘大道西华兴大厦 B901
电话：020-39885802 **传真：**020-39885802-803
网址：http://www.FDLock.com

文档名词约定

在本用户手册中，会把“广州飞盾电子有限公司”简写为“广州飞盾电子”或“飞盾电子”。

会把“NT132 多用户型多功能加密锁”简写为“NT132”或“NT132 多功能加密锁”或“NT132 加密锁”或“NT132 加密狗”(说明：加密锁又称为加密狗)。

会把“用户数据存储区”简写为“用户存储区”，或称为“用户数据存储空间”、“用户存储空间”。

会把“设备编号”称为“硬件 ID”。

会把“设置产品编号”称为“生成产品编号”，或“产生产品编号”。

会把“指示灯”称为“LED 指示灯”。

会把“用户分区”称为“扇区”。

“扇区密码”或“扇区的密码”就是此扇区的普通用户密码。

文档名词解释

加密锁——又称为加密狗，是一种插在计算机并行口/USB 接口上的软硬件结合的加密产品。加密锁一般都有几十、几百或几千字节的用户数据存储空间(非易失性存储器)，通常需要正确校验用户密码后才能对它进行读写操作，且密码校验失败次数是有限制的。

软件开发者可以通过接口函数和加密锁之间进行数据交换（即对加密锁进行读写操作），来检查产品配套的加密锁是否插在并行口/USB 接口上，如果没有插入或中途拔出了，则立即终止软件运行，或者使软件错误运行。软件开发者还可以在不同时间段，比如 1 月、2 月……，读取加密锁（用户数据存储空间的）不同位置的数据，并判断数据是否正确，然后控制软件是否正常运行；软件还可以根据数据的不同，来识别不同的操作员，然后相应的开放/禁止软件的某些功能。或者把程序的一小部分代码或运行参数写到加密锁上，运行前再从加密锁读出，并放到内存的相应位置上，如果没有相应的加密锁，程序将无法正确运行。或者直接用加密锁附带的工具加密自己的 EXE 文件（俗称“包壳”），如果没插相应的加密锁，则软件将不能正常执行。软件开发者可以在软件中设置多处软件“锁”，利用加密锁作为钥匙来打开这些“锁”，这样就使用加密锁来保护了开发者的软件成果。

修订记录

版本	日期	原因
V1.2	2009-12-15	原文档为“API 接口手册”，经过内容补充和修改后，改为“用户手册”来发布
V1.3	2010-2-21	修改页眉和页脚

目 录

1 产品概述

NT132 多用户型多功能加密锁(又称为加密狗)是专门针对广大软件开发公司而设计的一款基于 USB 接口的高性能、低价格、多用户分区的加密锁。该加密锁使用简单、操作方便，非常适用于软件加密保护、关键数据保存、安全系统身份认证领域，包括网站系统、OA 办公系统、信息查询系统、教学软件或其它行业软件等等。

NT132 加密锁无需安装驱动程序，内置 512 字节用户数据存储区(EEPROM)，划分为 4 个独立用户分区(每个分区 128 字节)；**每个用户分区都有独立的密码保护，如果一个软件使用一个用户分区，则一个 NT132 加密锁就能支持 4 种软件。如果只加密一个软件，软件在不同时间使用不同的密码，访问不同的存储分区，软件加密强度会更高。**用户数据存储区支持写次数 10 万次以上，读取次数不限；提供多操作员管理机制，方便用户对产品进行各种权限控制管理；支持自定义密码校验次数；支持自定义产品 OEM 显示信息；内置 48 位全球唯一硬件 ID，方便查询各个加密锁的使用情况或用于软件加密处理；自定义 1 至 128 位动态用户密码，加强解密难度；具有防复制功能；具有智能防跟踪功能；通信格式采用密文格式进行通信；支持多个设备在同一台计算机上使用。

NT132 多用户型加密锁是完全基于智能卡技术开发采用一颗高度集成化的智能芯片，芯片集成了其所有器件（包括 CPU、RAM、EEPROM 以及 USB 通讯模块），这极大的提高了产品的安全性和稳定性。每个芯片上都具有全球唯一的序列号(即硬件 ID)。

NT132 产品外观参考如 图 1.1 所示，尺寸为 50.0×18.0×10.5mm。

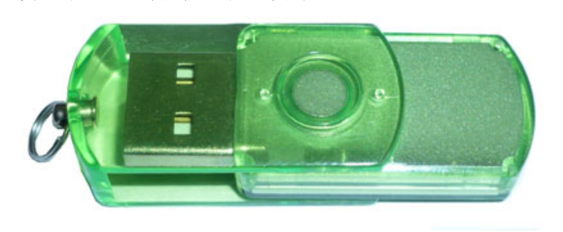


图 1.1 NT132 产品外观

1.1 功能特点

- 4 个独立用户分区，每个用户分区都有独立的密码保护；
- 无需安装驱动程序；
- 提供 48 位全球唯一硬件 ID；
- 支持在同一台 PC 上插入多把加密锁；
- 支持自定义产品 OEM 显示信息；
- 提供双操作员管理机制(普通用户和超级用户)；
- 超级用户密码最大长度为 128 位；
- 普通用户密码最大长度为 128 位；
- 支持自定义密码校验次数；
- 提供 512 字节用户数据存储空间 (即 4×128 字节)；
- 提供 64 位自定义产品编号；
- 设备与程序之间采用密文通信；
- 设备具有反跟踪功能；
- 设备上的数据采用 3DES、MD5 加密，即使剖开芯片也是读不到加密数据;
- 设备具有防复制功能；
- 工作电流 23mA/5V (典型值，指示灯为点亮状态)。

1.2 开发接口提供

- LIB 静态库方式；
- DLL 动态库方式；
- COM 组件方式；
- Active 控件方式；
- 外壳加密方式。

注：开发接口方式在不断增加中，请到我们网站 http://www.FDLock.com 查看最新支持的开发接口类别。如果没有你需要的接口，请您致电(020)32896151/39885802-805 给我们免费为您增加。

1.3 开发例子提供

- 应用程序：C#、CB5、Delphi、FoxPro6、JAVA、pb6、VB5、VC6、VBA、VB.net、Yin(易语言)**...**
- 网站网页：ASP、ASP.net、PHP、JSP
- 多媒体：Authorware6、Director9
- 第三方插件：Autocad2002

注：开发例子在不断增加中，请到我们网站 http://www.FDLock.com 查看最新例子。如果没有你需要的例子，请您致电(020)32896151/39885802-805 给我们免费为您增加。

1.4 操作系统支持

- 支持桌面 Windows 系列，如 Windows 98SE/Me/2000/XP/Server 2003/Vista；
- 支持嵌入式 Windows CE (即 WINCE)、Windows Embed XP；
- 支持 Linux；
- 支持 Mac OS。

2 产品专用术语

2.1 产品编号 PID

产品编号是一个长度为 64 位的编号，又称为 PID，主要是提供给软件开发厂商作为**区分不同的软件产品**使用的不同编号加密锁，比如软件 A，它只操作产品编号为 1234567890123456 的加密锁。产生产品编号的方法是：由软件厂商输入 1 至 56 个字符长度的产品密码(此密码只是用来生成产品编号，不会保存到加密锁中)，然后把产品密码送到加密锁中，由加密锁调用内部**硬件 3DES** 进行运算，生成 16 字节产品编号并设置到加密锁中，该过程是不可逆。也就是说只有知道产品密的的人才可以产生该产品编号，所以产品编号还具有一个功能是**防止加密锁被复制**。生成产品编号的权限为超级用户权限，所以要生成产品编号除了要知道产品密码外还要知道超级用户密码(16 字节)，所以该型号的加密锁**更难复制**。

软件编程时，PID 是以字符‘0’～‘9’、‘A’～‘F’的形式出现的，即一个字符只需 4 位值(16 进制)，所以 64 位就得到 64/4=16 个字符。

2.2 设备编号 DID

设备编号是一个长度为 48 位的编号，又称为 DID，由加密锁在制作过程中固化在加密芯片中的一组硬件编号(不可修改)。12 字节。每一个加密锁的设备编号都是不相同的，即全球唯一。该编号可以用作跟踪各个加密设备使用情况，如防代理商窜货等等功能；也可以用来作为加密算法中的一个加密条件，增强产品的安全性。

软件编程时，DID 是以字符‘0’～‘9’、‘A’～‘F’的形式出现的，即一个字符只需 4 位值(16 进制)，所以 48 位就得到 48/4=12 个字符。

2.3 产品密码

产品密码是用来产生产品编号的一个字符串，它的有效范围是 1～56 个字符，只要输入顺序及长度有所变化则产生的产品编号将会面目全非，所在请您在设置产品编号时一定要记住产品密码，否则我们也爱莫能助。

产品密码只是用来生成产品编号，不会保存到加密锁中。

2.4 超级用户密码 SPIN

超级用户密码是保障超级用户权限的一个密码，又称为 SPIN。由于这个密码的权限非常高级，一般应掌握在软件开发商核心管理人员手中。校验超级用户密码时，如果校验的密码不正确，加密锁只能进行各种匿名权限操作；**如果校验正确，则加密锁将把权限提升为超级用户权限**，即可进行超级用户的全部操作，如设置显示信息、设置产品编号、修改超级用户密码、修改普通用户密码，设置密码校验次数、读写数据等等。

超级用户密码最大长度为 128 位，即 16 字节，可以是任意 ASCII 码字符或汉字(一个汉字占 2 个字节)。

在任何权限状态下，超级用户密码均不可读出。

2.5 普通用户密码 UPIN

普通用户密码是保障普通用户权限的一个密码，又称为 UPIN。校验普通用户密码时，如果校验的密码不正确，加密锁只能进行各种匿名权限操作；**如果校验正确，则加密锁将把权限提升为普通用户权限**，可以修改普通用户密码，读写数据及各种匿名权限操作等等。

普通用户密码最大长度为 128 位，即 16 字节，可以是任意 ASCII 码字符或汉字(一个汉字占 2 个字节)。

NT132 加密锁划分为 4 个独立用户分区，每个用户分区都有独立的密码保护，即有 4 个普通用户密码。

在任何权限状态下，普通用户密码均不可读出。

2.6 密码校验次数

密码校验次数是校验密码时连续错误的最大次数，每校验错误一次则加密锁会对校验次数自动减一，当减至 0 次时则对该密码锁死(即不再允许校验此密码)；如果在未锁死之前校验密码正确，则加密锁会自动恢复校验次数为到原设置值。例如，设置用户密码校验次为 3 次，如果连续执行 3 次校验用户密码错误，则加密锁就把用户密码锁死。如果用户在第 2 或第 3 次时校验密码正确，则把密码校验次数又恢复到 3 次。

当普通用户密码锁死后，可以先校验超级用户密码，然后再设置相应的校验次数即可恢复。如果超级用户密码锁死，则该加密锁只能报废。当密码校验次数设为 0 时，则表示为不限制校验错误次数，即是可以任意校验。

NT132 加密锁划分为 4 个独立用户分区，每个用户分区的密码校验次数都是可以设置的，且可以互不相同的。

2.7 用户权限

用户权限是加密锁对不同用户的权力分配/操作限制。加密锁提供多操作员管理机制，类别分为普通用户和超级用户，也称为普通用户权限和超级用户权限。超级用户的权限最高，可以进行设置显示信息、设置产品编号、修改普通用户密码，设置密码校验次数、读写数据等操作。普通用户的权限则只能进行读写数据、修改普通用户密码及各种匿名权限等操作。

加密锁上电后就处于匿名权限状态；当校验普通用户密码正确时，加密锁将把权限提升为普通用户权限；当校验超级用户密码正确时，加密锁将把权限提升为超级用户权限。如果校验失败，加密锁将返回到匿名权限状态。

在第 6 章的 API 函数说明中，每一个 API 函数都会指明其用户权限类别，即表示在什么用户权限下可调用此 API 函数。

2.8 产品显示信息

产品显示信息，是用来指示产品厂家、厂家网址或产品型号的字符信息，在**第一次使用加密锁**时WINDOWS右下角会弹出的此字符信息，参考如 图 2.1 所示。另外，在设备管理器中对应的“USB人体输入设备”属性页中，也会有此字符信息，参考如 图 2.2 所示。我们在这里为软件开发商提供显示加密锁的个性信息或广告信息，而不是我们生产厂商的信息，这为软件开发商提供了更好的OEM服务。

显示信息为 17 个汉字或 17 个 ASCII 码字符，请在输入显示信息时不要超过该长。



图 2.1 插入时显示信息

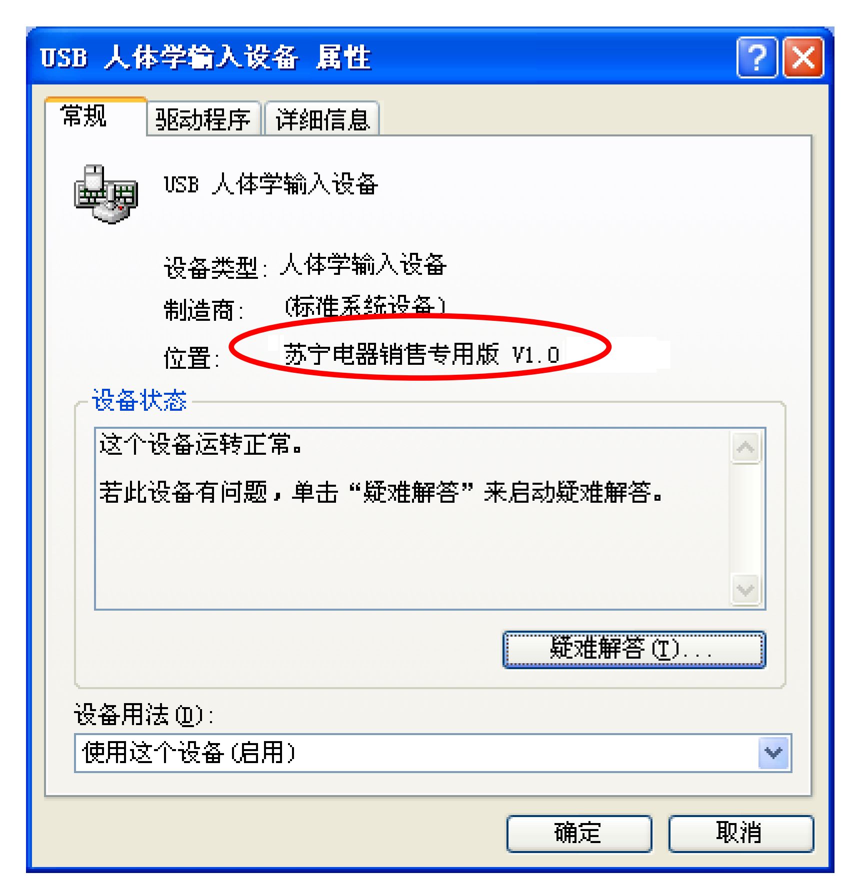


图 2.2 设备属性页显示信息

2.9 用户分区

NT132 加密锁将 512 字节用户数据存储空间划分成 4 个独立的用户分区(后面又称为扇区)，每个分区都有独立的密码保护，参考如 图 2.3 所示。

只有校验成功相应扇区的普通用户密码后，或者校验成功超级用户密码后，才能访问此扇区的 128 字节存储空间，操作地址统一为 0～127。软件开发商可以在加密锁不同的扇区上写入一些不相同的校验数据，设置互不相同的密码，然后一个软件使用一个用户分区，使一个 NT132 加密锁就能支持 4 种软件。如果只加密一个软件，软件在不同时间使用不同的密码，访问不同的存储分区，软件加密强度会更高。

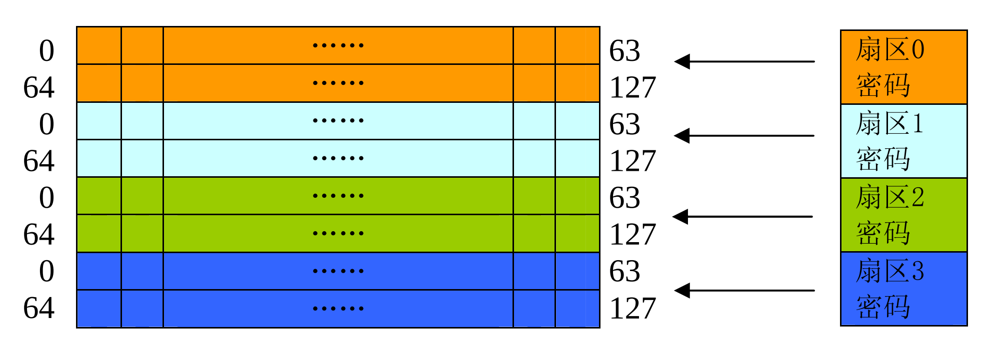


图 2.3 用户数据存储空间示意图

2.10 当前扇区

NT132加密锁具有4个独立的用户扇区(即分区)，同一时刻只能访问一个扇区的存储空间，即当前扇区。如何使某个扇区变为当前扇区呢？首先要选择此扇区，然后校验此扇区的密码(普通用户密码)或超级用户密码，校验成功后，此扇区即变为当前扇区。

注意：超级用户权限状态下，如果要修改某一扇区的密码，也需要先选择此扇区。

3 产品出厂默认设置

NT132 加密锁出厂默认设置如下：

- 产品编号**(PID)**：16 个字符‘0’，即是“0000000000000000”；
- 超级用户密码**(SPIN)**：16 个字符‘8’，即是“8888888888888888”；
- 各分区的普通用户密码**(UPIN)**：16 个字符‘8’，即是“8888888888888888”；
- 产品显示信息：“www.FDLock.com”；
- 超级用户密码校验次数：为 0，即不限次数；
- 各分区的普通用户密码校验次数：为 0，即不限次数；
- 数据存储空间：全为 0xFF。

注意：产品编号是用来识别加密锁的唯一标识，如果没有正确的产品编号，则无法打开和操作加密锁。所以用户在产生新的产品编号时，一定要记住新的产品编号和产品密码。

4 快速入门

如果您是第一次使用NT132加密锁，请先阅读本手册的第1～3章的内容，以便于对此产品及相关术语有所了解。

本章内容分为两部分，第一部分是介绍使用NT132的Demo(即演示例子)来进行各种操作，可以使您对NT132有基本的、直观的认识；第二部分介绍如何使用加密锁来保护软件，帮助您对使用加密锁来加密软件的基本原理和思路有所了解。

4.1 演示例子的使用

(1) 获得开发套件

NT132开发套件参考如图4.1所示，用户可以通过购买来获得。



图 4.1 NT132 开发套件

(2) 将加密锁插入到**PC**

将NT132插入到PC机的USB接口，参考如图4.2所示，此时NT132的指示灯会点亮(在NT132的中间部位可以看到)。

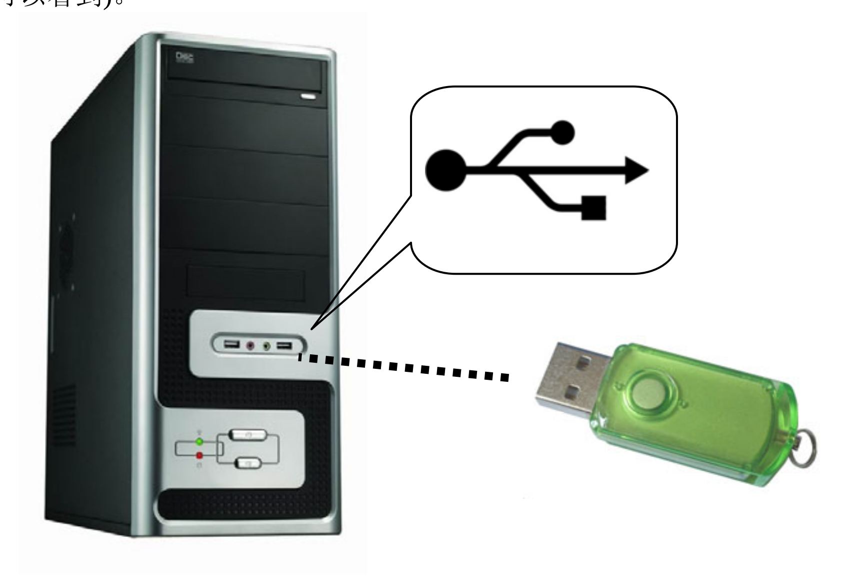


图 4.2 将加密锁插入 PC 机的 USB 接口

(3) 运行加密锁演示程序

将NT132开发套件的产品光盘放到PC机的光驱中，然后打开光盘“\快速入门”目录，双击运行“NT132_DEMO.exe”，将会出现如图4.3所示的主界面。

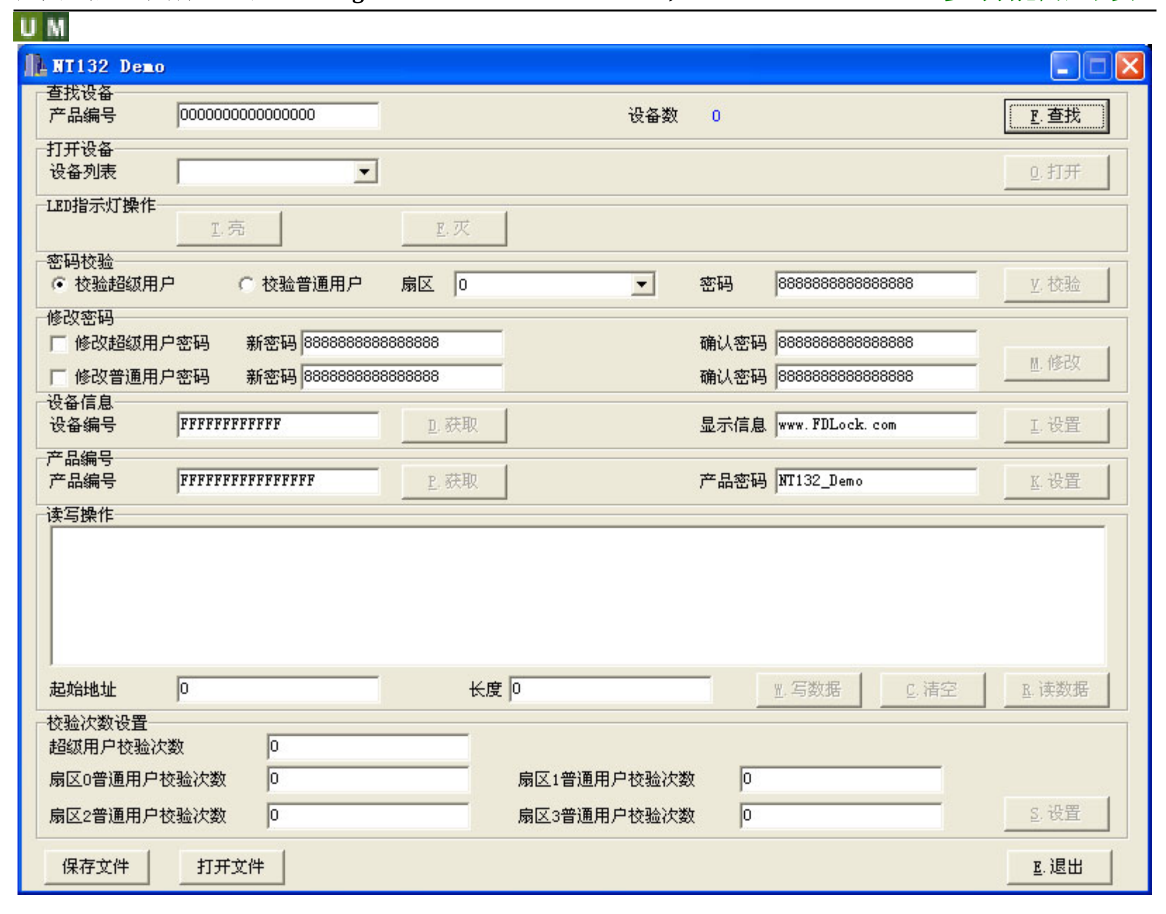


图 4.3　NT132 Demo 主界面

(4) 查找设备

使用NT132加密锁的第一步是查找指定产品编号的加密锁。在NT132 Demo主界面的“产品编号”栏中输入产品编号(16个‘0’～‘9’、‘A’～‘F’字符)，如16个字符‘0’(加密锁的出厂默认设置值)，如图4.4所示，然后单击右边的“查找”按钮即可。

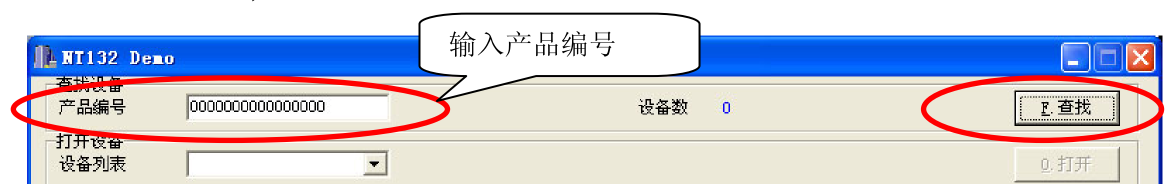


图 4.4　查找指定产品编号的加密锁

若找到相应产品编号的加密锁，则“查找”按钮的左边显示当前找到的设备数目(如果插入3个相同产品编号的加密锁，则设备数为3)，如图4.5所示。如果没有找到，则会弹出如图4.6所示的错误提示。

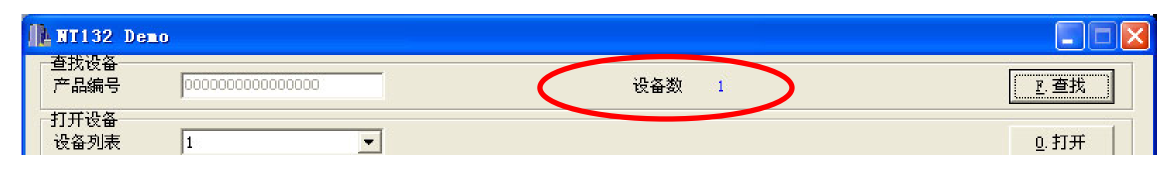


图 4.5　找到相应的加密锁



图 4.6 没有找到加密锁的提示

(5) 打开设备

第二步是打开设备，即打开加密锁。在NT132 Demo主界面的“设备列表”栏中选择要打开的设备，如果前面查找到的设备数为1，则只能选择“1”，如果查找到的设备数为N，则可以选择1～N，然后单击右边的“打开”按钮即可，如图4.7所示。

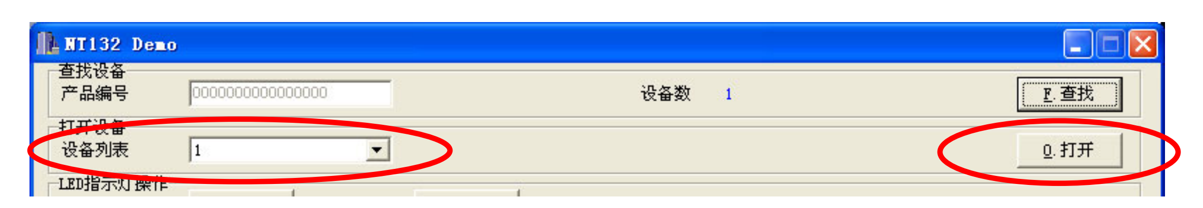


图 4.7 选择并打开设备

(6) 匿名权限状态下的各种操作

在没有成功校验普通用户或超级用户密码之前，加密锁处于匿名权限状态，此时可以进行一些简单的操作。

- LED指示灯操作

单击“LED指示灯操作”栏的“亮”按钮，即可控制NT132加密锁的指示灯点亮(在NT132的中间部位可以看到)；而单击“灭”按钮，即可控制NT132加密锁的指示灯熄灭，如图4.8所示。

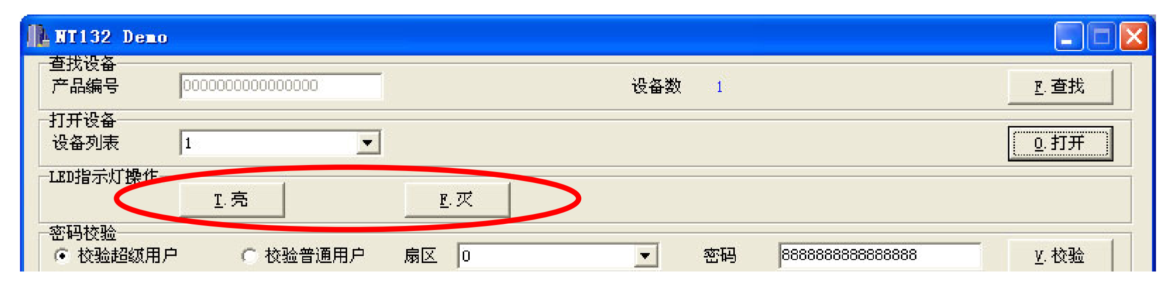


图 4.8 指示灯控制

- 获取设备编号DID

单击“设备编号”右边的“获取”按钮，即可取得当前加密锁的设备编号，即DID，参考如图4. 9所示。

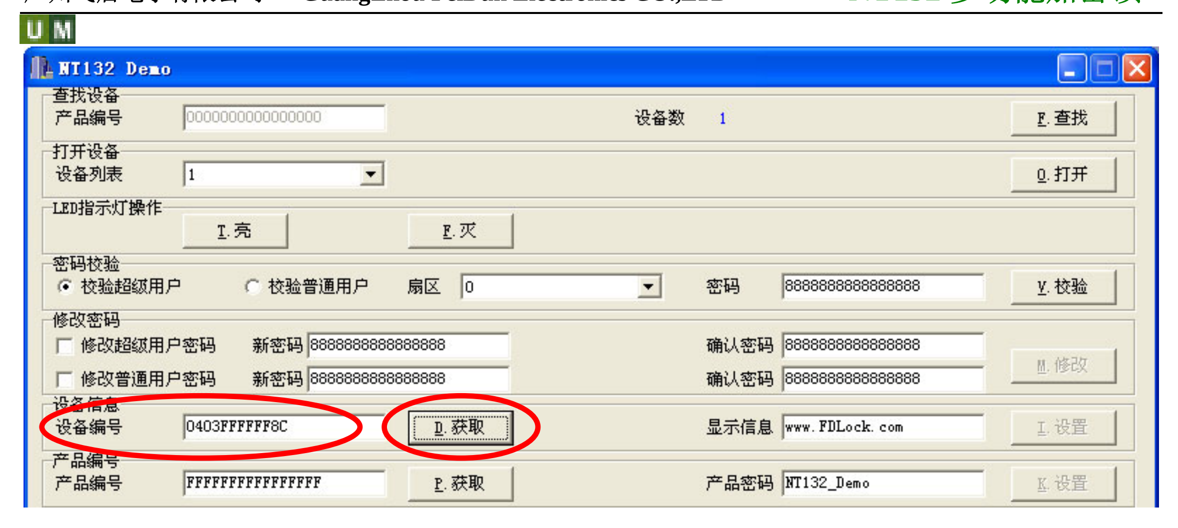


图 4.9 获取设备编号

(7) 校验普通用户密码

匿名权限状态下，对加密锁的操作是有限的，而且不能对用户数据存储区进行读写操作，所以要通过校验普通用户密码，切换到普通用户权限。在NT132 Demo的主窗口中，在“密码校验”栏中选中“校验普通用户”，然后在其右边的“扇区”栏选择要操作的扇区，比如选择3，接着在“密码”栏中输入此扇区的密码，如16个字符‘8’(加密锁的出厂默认设置值)，如图4.10所示。输好密码后，单击右边的“校验”按钮，即执行校验普通用户密码操作，如果校验成功则提示“校验密码成功”，否则提示“密码错误”。 校验密码成功后，加密锁即切换到普通用户权限状态下。

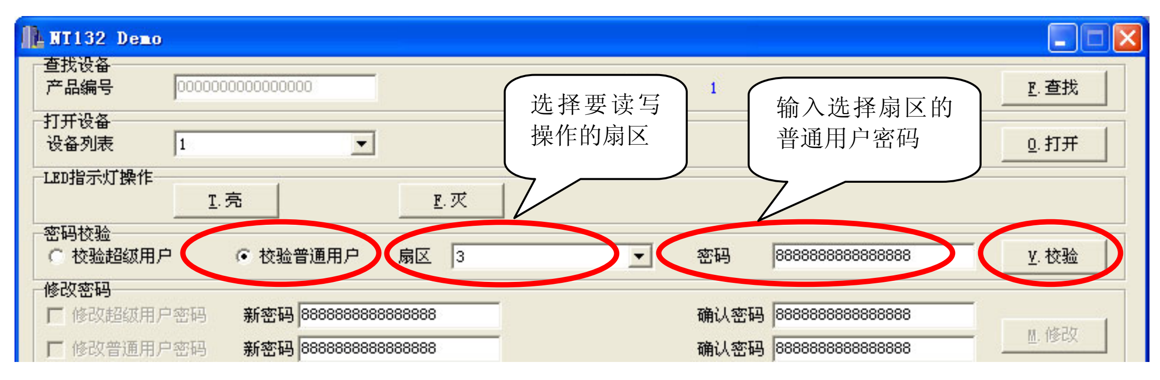


图 4.10 校验普通用户密码操作

(8) 普通用户权限下的各种操作

● 修改普通用户密码

在普通用户权限状态下，可以修改**当前扇区的**普通用户密码(比如前面是选择扇区3来校验密码，则当前扇区就是扇区3)。首先要选中“修改普通用户密码”，然后在其右边的“新密码”、“确认密码”栏中输入相同的新密码，最后单击右边的“修改”按钮，即可完成密码修改，参考如图4.11所示。

新密码可以是1～16个ASCII码字符，也可以是汉字(一个汉字占2个字节)。

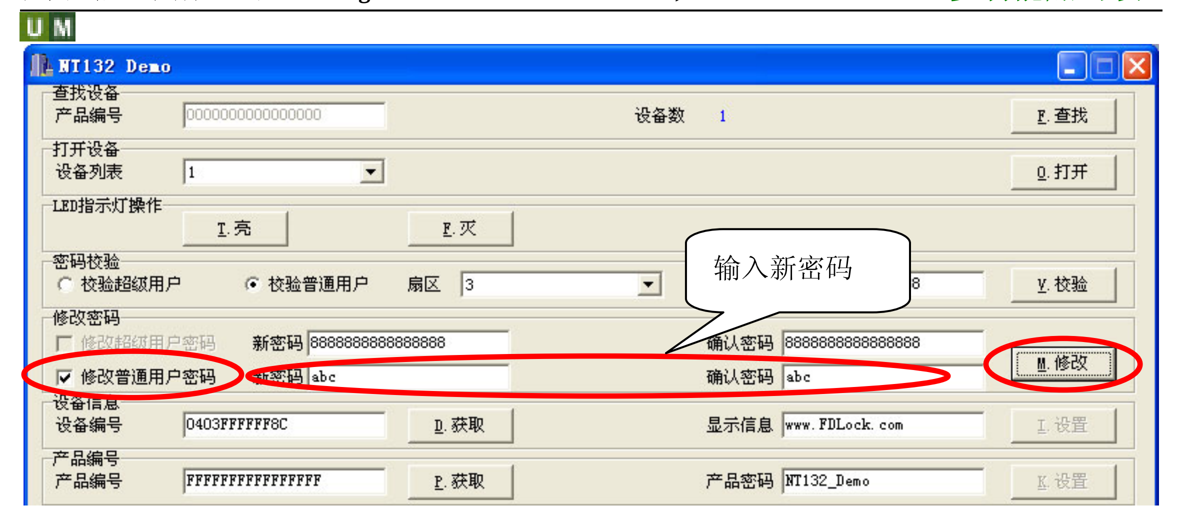


图 4.11 修改普通用户密码

- 读写用户数据存储区

在普通用户权限状态下，可以对**当前扇区的**用户数据存储区进行读写操作。如图4.12所示，首先在“起始地址”栏中写入要读数据的起始地址，NT132的每个扇区为128字节用户存储空间，所以地址范围是0～127。然后在“长度”输入要读取的数据长度，即读取多少字节数据。如果要读完全部数据，则起始地址要设为0，长度要设为128。最后单击“读数据”按钮，即可读取相应的数据，并在“数据显示/编辑区”中显示出来(**如果读取到的数据不是可显示字符的值，则看到不数据，比如0xFF就不能显示**)。

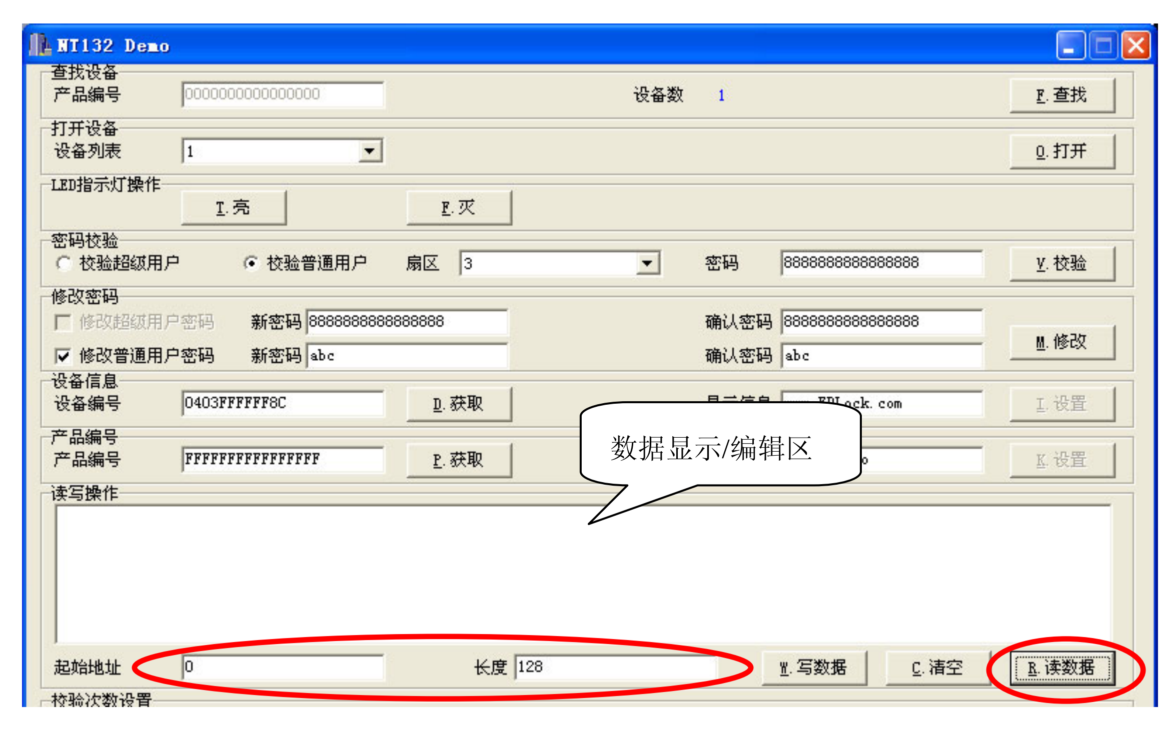


图 4.12 用户数据存储区读操作

若要修改某地址开始的多字节数据，首先点击“清空”按钮，此时“数据显示/编辑区”的内容将被清空；然后在“数据显示/编辑区”中输入要修改的数据，比如“ABCDEFGHIJKLMNOPQRSTUVWXYZ”，此时“长度”栏会自动显示已输入字符的个数；接着，在“起始地址”输入要修改(即写入)数据的开始地址；最后单击“写数据”，即可把输入的数据写到指定的地址空间上，参考如图4.13所示。

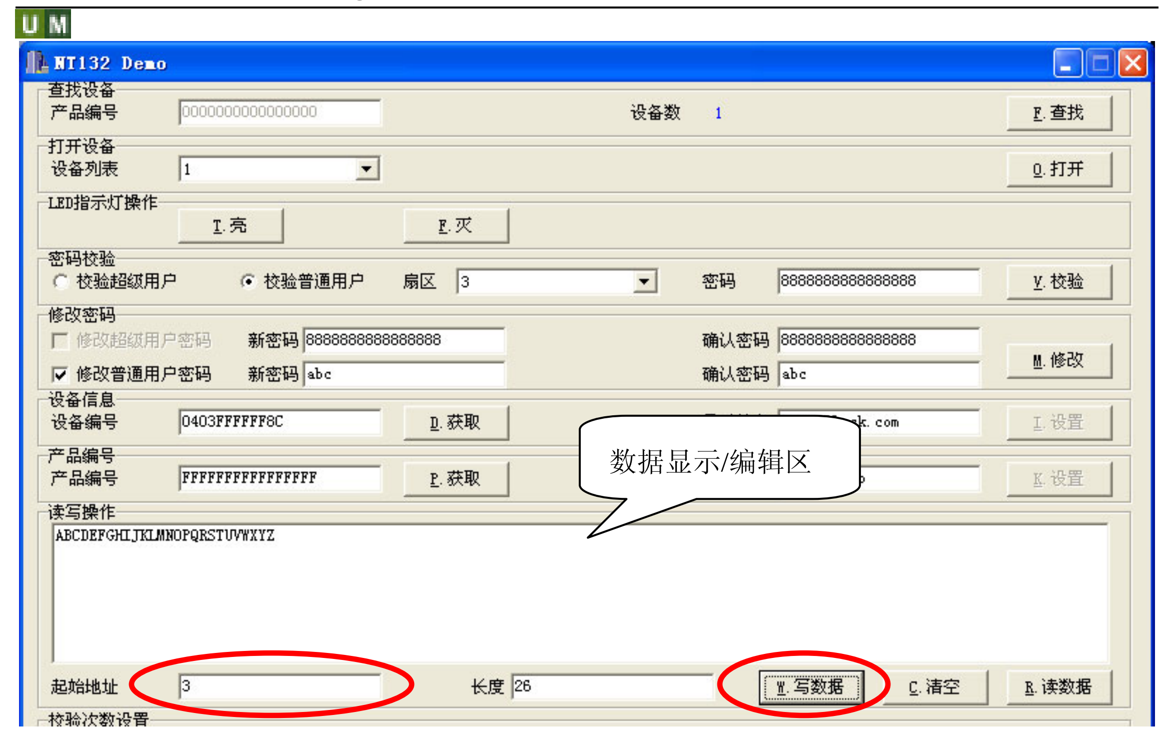


图 4.13 用户数据存储区写操作

(9) 校验超级用户密码

在NT132 Demo的主窗口中，在“密码校验”栏中选中“校验超级用户”，然后在其右边的“扇区”栏选择要操作的扇区，比如选择3，接着在其右边的“密码”栏中输入密码，如16个字符‘8’(加密锁的出厂默认设置值)，如图4.14所示。输好密码后，单击右边的“校验”按钮，即执行校验超级用户密码操作，如果校验成功则提示“校验密码成功”，否则提示“密码错误”。校验密码成功后，加密锁即切换到超级用户权限状态下。

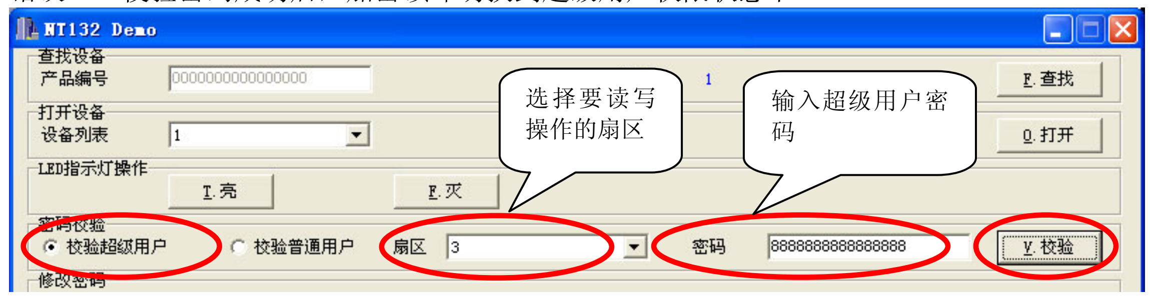


图 4.14 校验超级用户密码操作

(10) 超级用户权限下的各种操作

在超级用户权限状态下，可以修改超级用户密码、(当前扇区的)普通用户密码以及密码校验次数，可以设置产品显示信息，可以产生新的产品编号，可以读写用户数据存储区等等。

- 修改超级用户密码

首先要选中“修改超级用户密码”，然后在其右边的“新密码”、“确认密码”栏中输入相同的新密码，最后单击右边的“修改”按钮，即可完成密码的修改，参考如图4.15所示。

新密码可以是1～16个ASCII码字符，也可以是汉字(一个汉字占2个字节)。

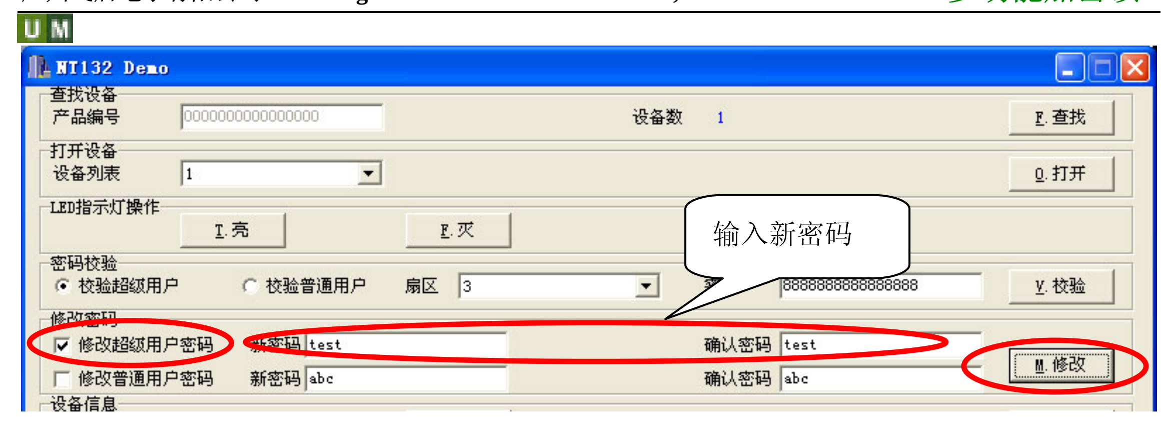


图 4.15　修改超级用户密码

注意：超级用户权限是具有最高的操作权限，请一定要记住新的超级用户密码，否则以后将不能修改加密锁的相关设置。

- 读写用户数据存储区

在超级用户权限状态下，可以对**当前扇区的**用户数据存储区进行读写操作。具体参考普通用户权限下的读写用户数据存储区操作。

- 设置产品显示信息

在“设备信息”栏的“显示信息”处输入要写显示的字符串，然后单击其右边的“设置”按钮即可，参考如图4.16所示。

显示信息为17个汉字或17个ASCII码字符，请在输入显示信息时不要超过该长度。

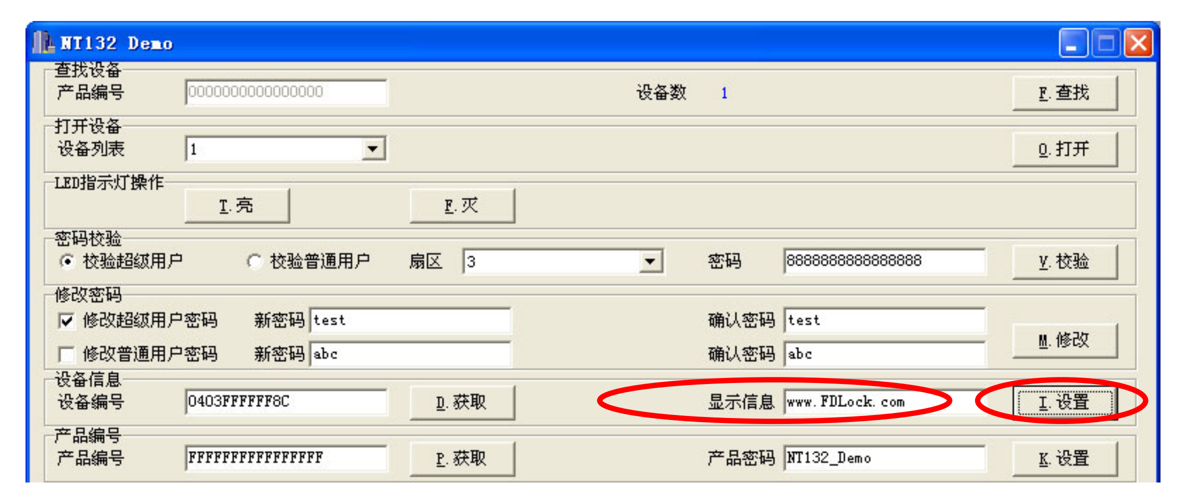


图 4.16　设置产品显示信息操作

- 产生新的产品编号

在“产品编号”栏的“产品密码”处输入用于生成产品编号的产品密码，并请记住此产品密码，然后单击其右边的“设置”按钮，加密锁即会产生新的产品编号(并保存在加密锁内)，同时在“产品编号”处会显示产生的产品编号，请记住此产品编号，参考如图4.17所示。

产品密码可以是1～56个ASCII码字符，也可以是汉字(一个汉字占2个字节)。

说明：为了防止用户在使用NT132 Demo软件时误操作，产生了新的产品编号，而又没有记住生成的产品编号，导致下次使用时无法找到设备(即加密锁)，所以NT132 Demo软件中设置产品编号的“设置”按钮的功能被删除了。

如果用户需要使用此功能，请将产品光盘“\example_DLL”整个目录复制到D:\中，并去掉整个目录的只读属性；然后使用相应的集成开发环境去打开D:\example_DLL目录下的工程，

比如使用Visual C++ 6.0，则打开D:\example_DLL\VC6\NT132_DEMO目录下的工程；接着找到"产品编号"栏下的"设置"按钮的处理代码，把原来注释的代码取消注释；保存修改的文件，重新编译工程，最后运行程序即可。

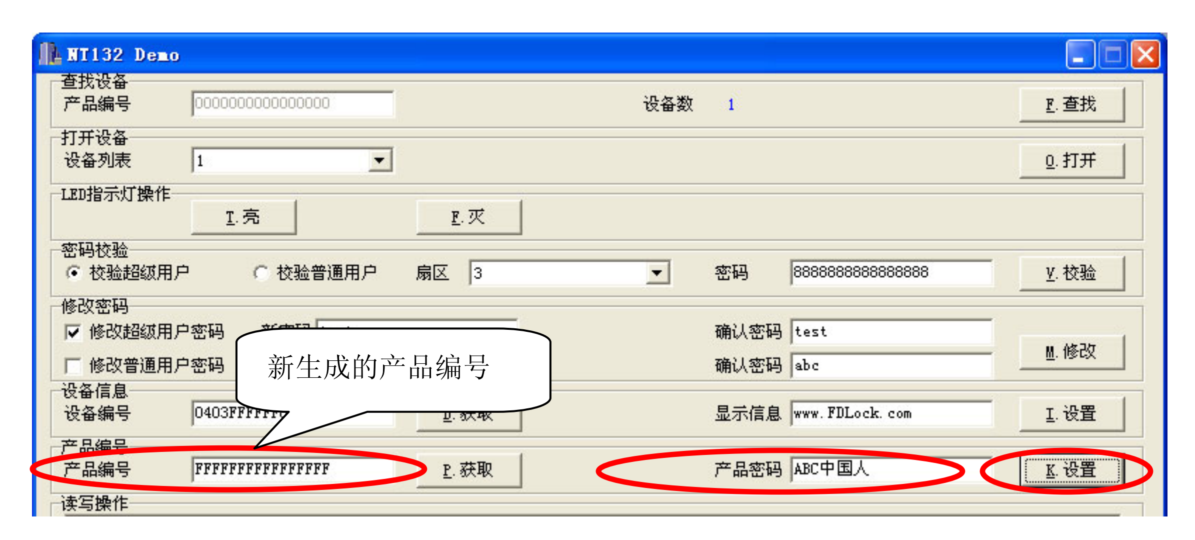


图 4.17 产生新的产品编号

注意：产品编号是用来识别加密锁的唯一标识，如果没有正确的产品编号，则无法打开和操作加密锁。所以用户在产生新的产品编号时，一定要记住新的产品编号和产品密码。

- 设置密码校验次数

在"校验次数设置"栏下有5个设置栏，包括1个超级用户密码校验次数设置栏，4个扇区的普通用户密码校验次数设置栏。

在"超级用户校验次数"处输入超级用户密码的密码校验次数，值为0～255，其中0表示不限次数；在扇区0～3的普通用户校验次数处，分别输入4个扇区的普通用户密码的密码校验次数，值为0～255，其中0表示不限次数。

修改好密码校验次数后，单击右边的"设置"按钮即可，参考图4.18所示。

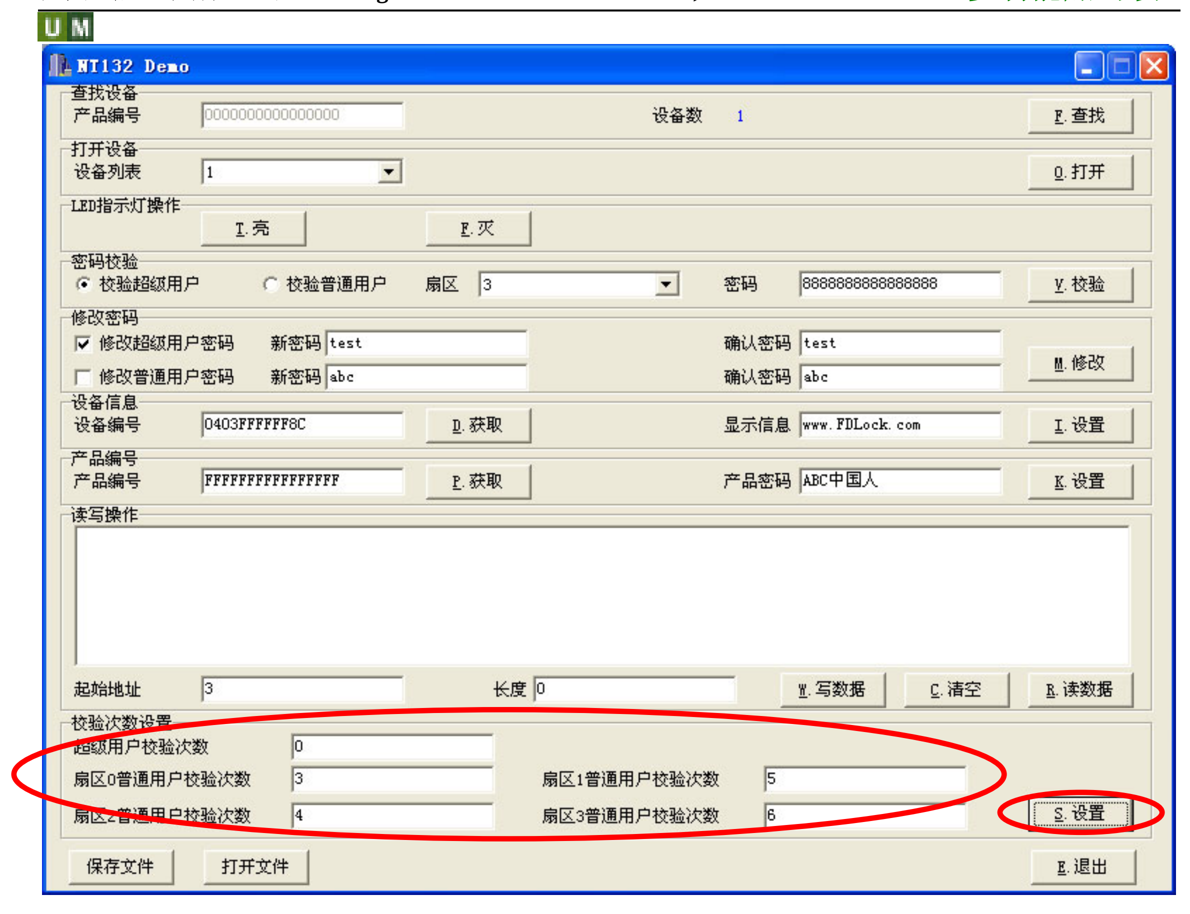


图 4.18 设置密码校验次数

(11) 其它操作

- 保存文件和打开文件

在NT132 Demo主界面的最下面，有一个“保存文件”和一个“打开文件”按钮。“保存文件”是将当前NT132 Demo主界面的设置数据(包括各种密码、“数据显示/编辑区”中的数据等等)保存成为一个*.NTF文件；“打开文件”则是将*.NTF文件中的设置数据加载进来。

4.2 如何使用加密锁来保护软件

使用加密锁来保护软件(即对软件进行加密)，主要是保护软件不被非法复制和使用，非授权访问或操作。使用加密锁对软件进行加密的形式是多种多样的，但最基本的原理是软件开发商编写的程序通过API接口函数和加密锁之间进行数据交换（即对加密锁进行读写操作），检查产品配套的加密锁是否插在USB接口上，如果没有插入或中途拔出了，则立即终止软件运行，或者以错误的方式运行，示意图如图4.19所示。由于加密锁具有各种功能特性，如设备编号、产品编号、用户权限(密码)管理、用户数据存储区、密文通信、防反跟踪功能等等，使它具有不可复制性，所以软件也就具有不可复制性。

另外，软件保护的强度不仅仅依赖加密锁本身，很大程度也和软件开发者如何使用加密锁相关，所以软件开发者要充分利用加密锁的功能，在软件中设置多处不同形式的“锁”，利用加密锁作为钥匙来打开这些“锁”。

(1) 加密锁的基本用法

加密锁一般都有几十、几百或几千字节的用户数据存储空间(非易失性存储器)，通常需要正确校验用户密码后才能对它进行读写操作(**所以软件中一般都包含有普通用户密码，但不要有超级用户密码**)。软件开发者可以在不同时间段，比如1月、2月、3月……，读取加密锁用户数据存储空间的不同位置的数据(这此数据是由软件开发商预先写进去的)，并判断数据是否正确，然后控制软件是否正常运行。

软件开发者可以让软件根据加密锁的存储数据的不同，来识别不同的操作员，然后相应的开放/禁止软件的某些功能。通过使用加密锁来替换传统的用户名和密码，保证了系统的登录安全。

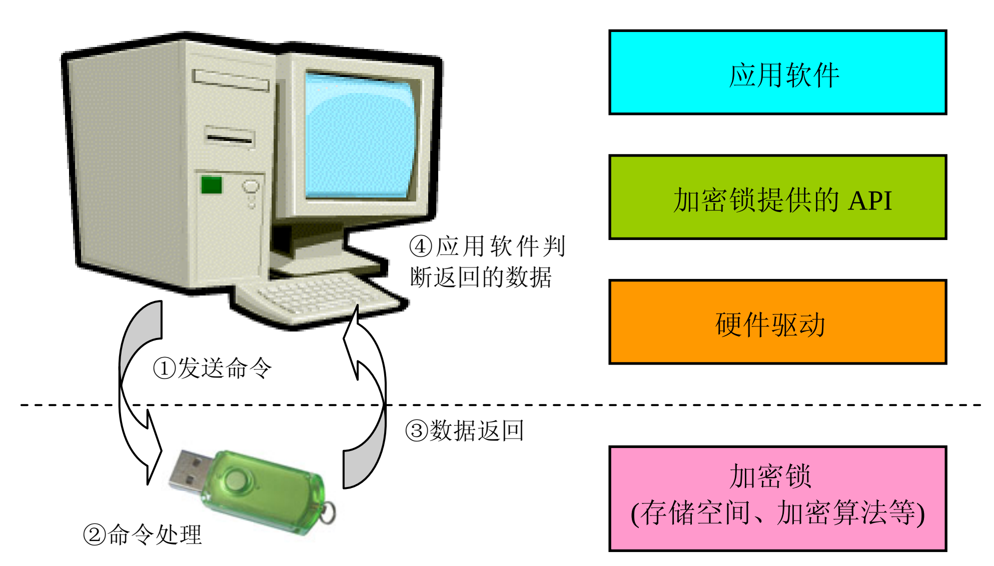


图 4.19 加密锁应用的基本原理

软件开发者可以把程序的一小部分代码或运行参数写到加密锁上，运行前再从加密锁读出，并放到内存的相应位置上，如果没有相应的加密锁，程序将无法正确运行。

软件开发者可以使用设备编号(即DID)和数据进行运算，再根据运算结果来控制程序是否运行。

以上介绍的都是调用API来加密软件的方式，这需要结合软件的源程序来进行。另外，还可以直接用加密锁附带的工具来加密EXE文件（即外壳加密），如果没插相应的加密锁，则软件将不能正常执行。

(2) 提高软件的加密强度

对于调用API来加密软件的方式，软件开发者可以在程序设计上做一些处理，以提高软件的加密强度，参考如下：

- 访问加密锁之后不要立即做判断，判断加密锁不正确后，不要立即输出提示信息，或者干脆不输出提示信息，或者运行一些无用的代码；
- 在程序的各个部分插入校验算法的代码，不要简单的、统一的调用同一个函数来校验，这样就增加程序代码的复杂度；
- 校验代码插入程序中的地方越多，破解难度越大，软件就越安全；
- 充分利用加密锁的功能，利用不同的时间、存储空间，尽量使用多种不同的校验方式。这样的话，只要破解者没有枚举/探测完所有的访问方式和校验方式，软件就不会被真正破解；
- 可以对加密锁进行一些随机性的、无关的操作，如读写没有用到存储空间；
- 重要的字符串在程序中尽量不要以明文出现(比如要通过2个数组相“异或”才得到)；
- 软件中不要出现和使用超级用户密码；
- 普通用户密码和产品编号不要以字符串形式出现，应使用多个数组来分开保存，甚至可以通过2个数组相“异或”才得到；
- 最好使用动态密码，即每个加密锁的普通用户密码与它的DID有关联；
- 最好能检查EXE或DLL文件的完整性。

5　开发流程

本章内容将会介绍NT132加密锁的API接口调用方式和基本操作流程，然后以Visual C++ 6.0为开发平台，分别使用LIB静态库方式和DLL动态库方式举例，实现对NT132的基本操作(没有涉及软件保护方面的设计)。例子程序中的代码简单明了，也没有使用任何特殊技巧，所以使用其它语言的软件开发人员都可以把它作为参考。

说明：例子中所用到的API函数的介绍，请参见第6章。

5.1　API接口调用方式

NT132多用户型多功能加密锁提供有多种API接口调用方式，具体如下：

- **LIB静态库方式**

该方式的好处是编译生成的EXE文件可独立运行，无需附带其它文件，也无需向系统注册相关文件。目前只支持 **Visual C++** 和 **Boland C++**两种开发语言。

使用LIB静态库的操作方法是：在开发软件时，把NT132L.LIB及NT132.H 文件复制到工程目录下，然后在工程设置中包含它们，这样就可以在程序中调用相应的API函数了。

- **DLL动态库方式**

该方式是大多数编程序语言都支持的一种开发方式，用户只需在程序中进行声明，然后再调用相应的API函数即可，程序在运行时必须保证NT132.DLL 文件在系统目录下或在此EXE文件的同一目录下。

- **COM 组件方式**

该方式也是绝大大多数编程语言都支持的开发方式，**COM**组件提供了相应的API接口，程序运行时必须保证NT132C.DLL组件已存在，并且已注册。

- **Active控件方式**

该方式也是绝大大多数编程语言都支持的开发方式，**OCX**控件提供了相应的API接口，程序运行时必须保证NT132.OCX控件已存在，并且已注册。

5.2　API调用操作流程图

不管使用哪一种API接口调用方式，其操作流程图都是相似的，参考如 图5.1所示，具体代码请参考产品配套光盘上的示例程序。**图中的“设备”，指的是加密锁，后面的说明中也有这样的表述。**

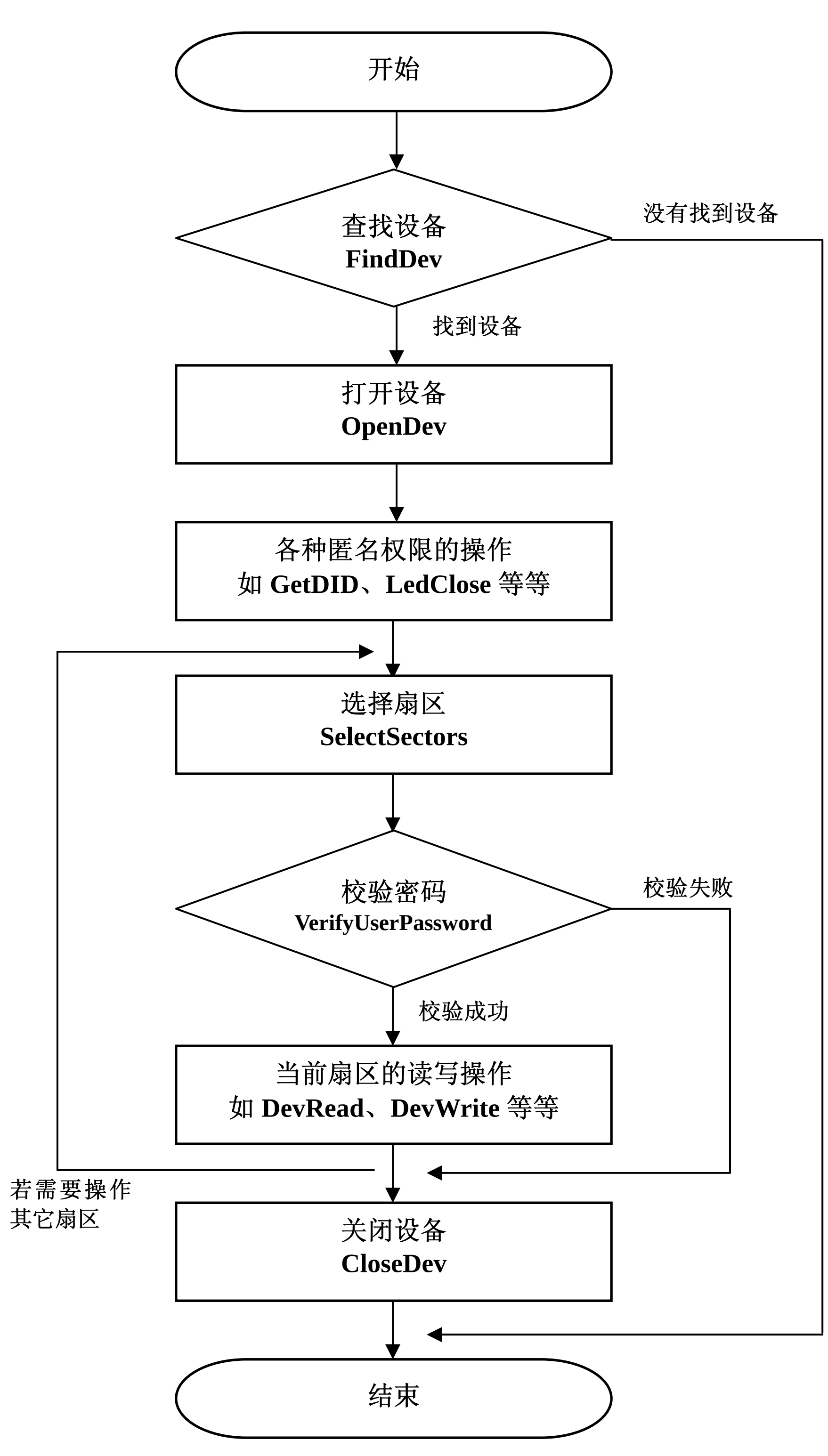


图 5.1 API 函数调用总体流程图

5.3 基本应用示例

5.3.1 基于 LIB 静态库方式

(1) 启动Visual C++ 6.0，新建立一个基于对话框的工程NT_TEST1，工程保存路径为D:\，如图5.2、图5.3所示。

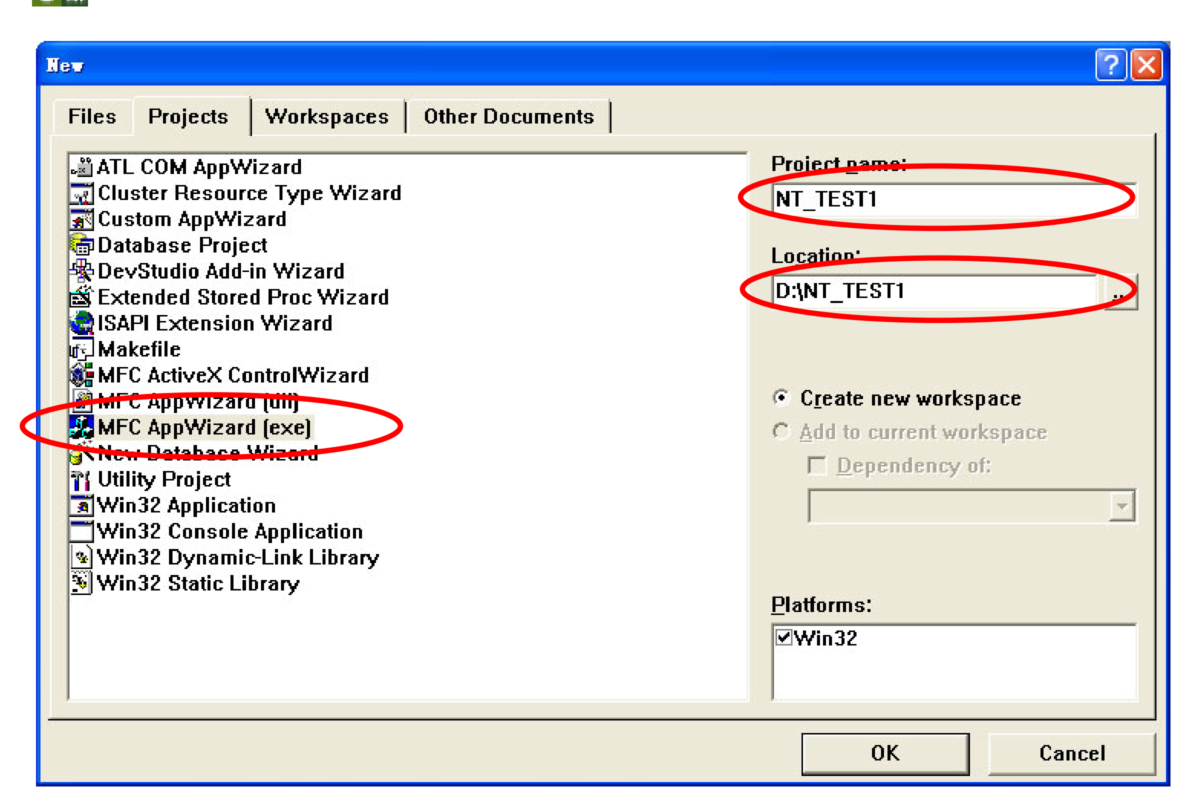


图 5.2 新建立工程 NT_TEST1

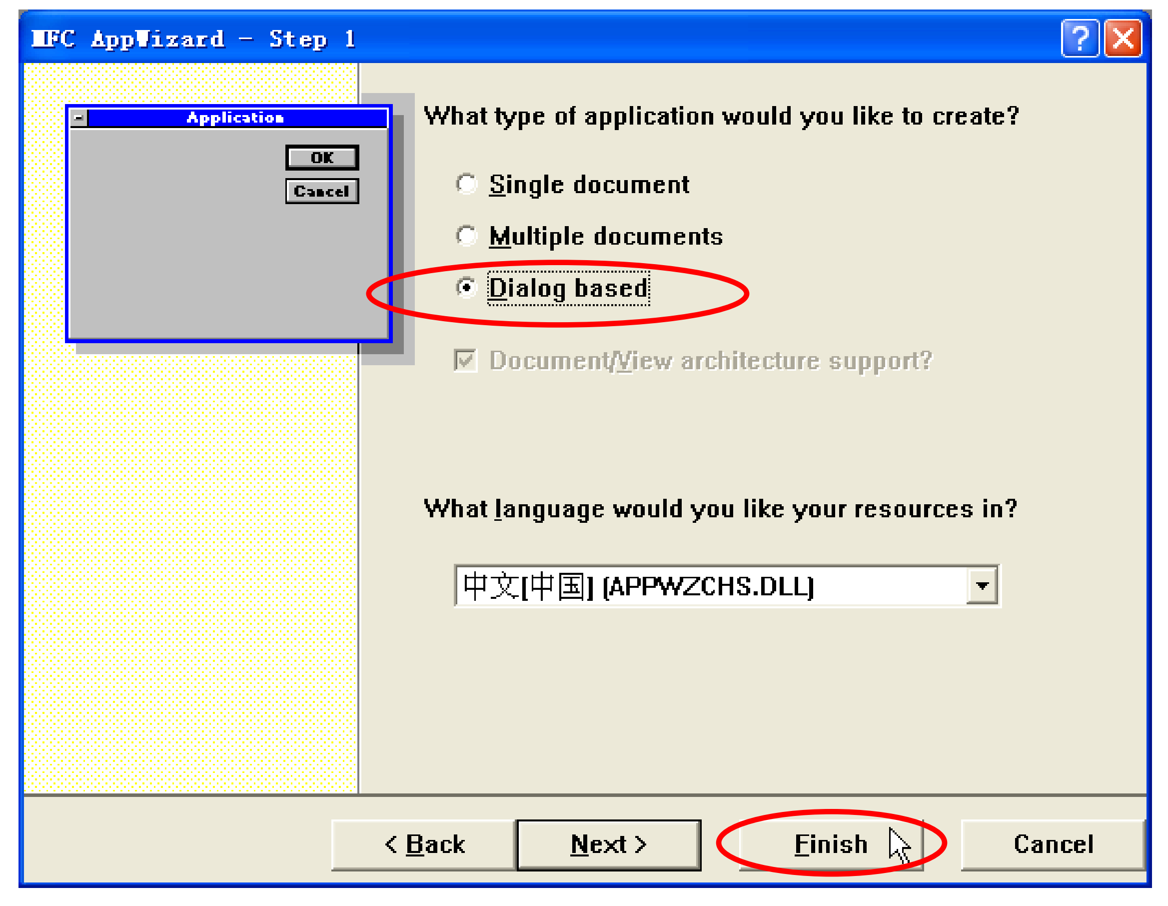


图 5.3 设置 NT_TEST1 为基于对话框的应用

(2) 将产品配套光盘“\Lib\NT132_VC”目录下的3个文件复制到D:\NT_TEST1目录下，结果如图5.4所示。

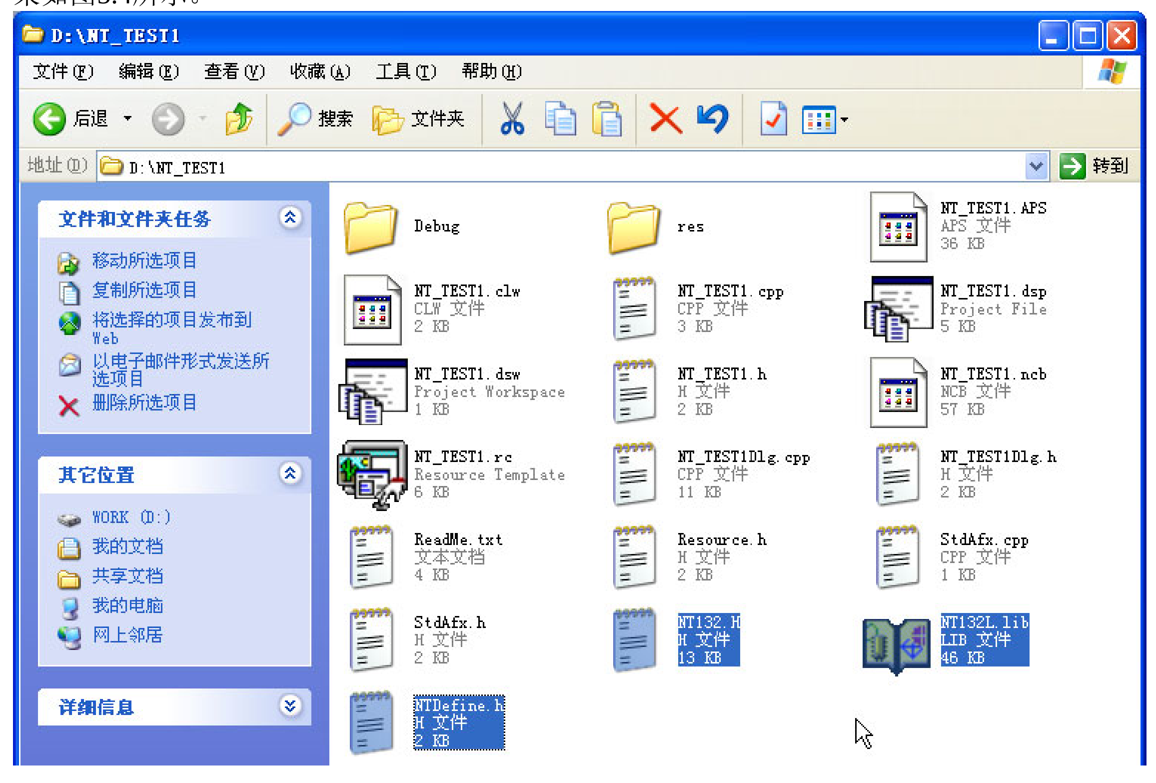


图 5.4 复制 NT132 的 LIB 文件

(3) 在FileView页面中右击“NT_TEST1 files”，在弹出的浮动菜单中选中“Add Files to Project…”，再将工程目录下的NT132L.lib文件添加进工程，如图5.5、图5.6、图5.7所示。

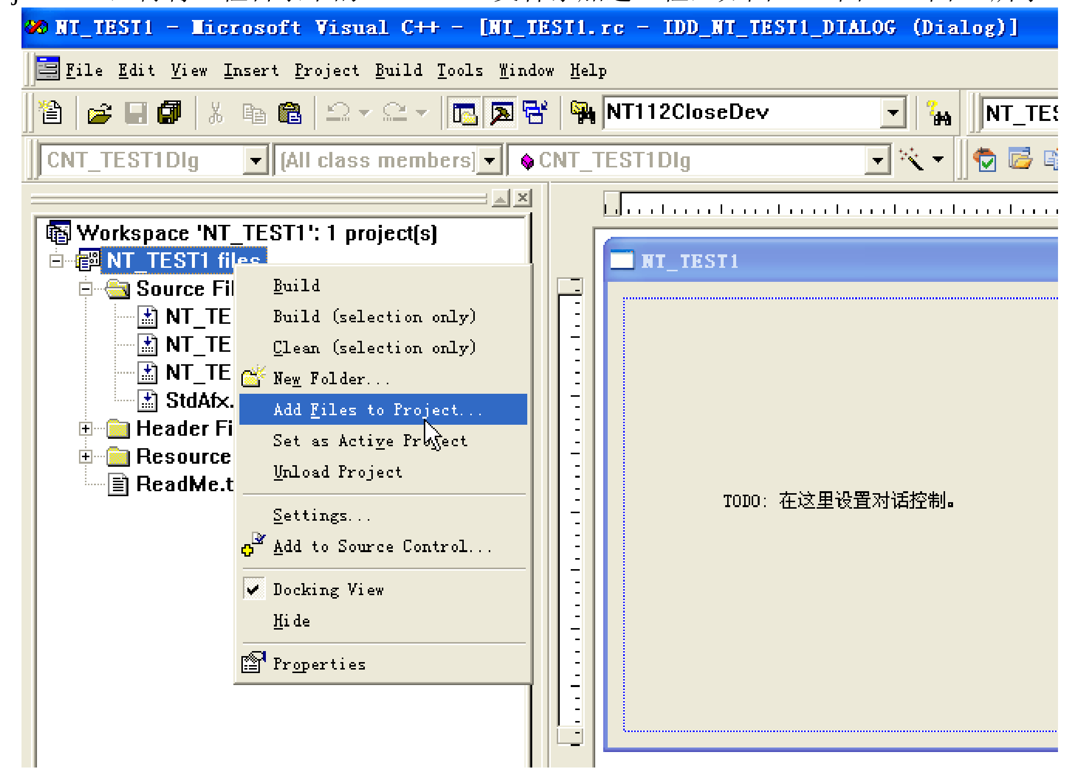


图 5.5 选择 Add Files to Project 子菜单

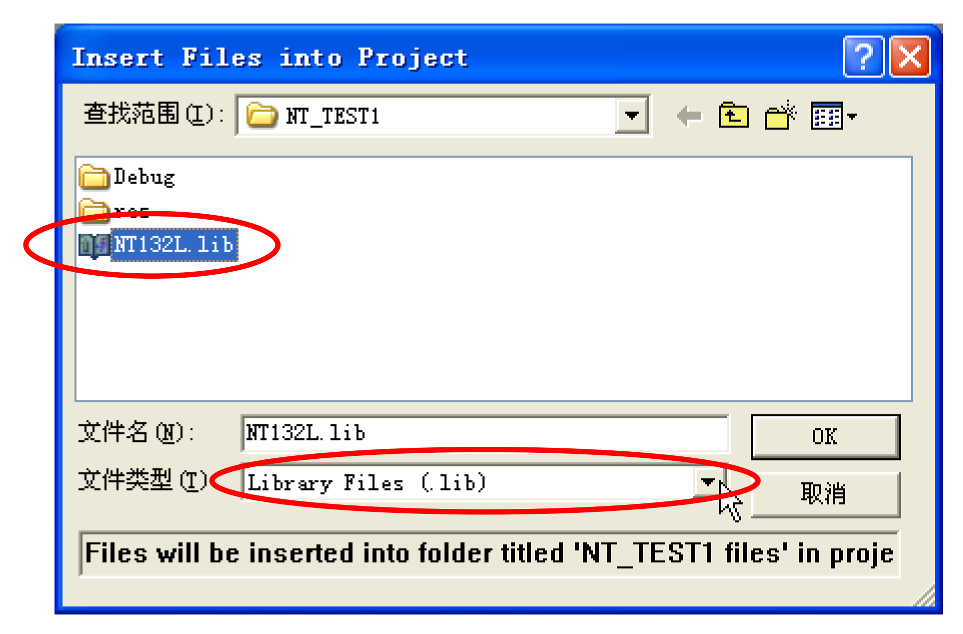


图 5.6 选择工程目录下的 NT132L.LIB 文件

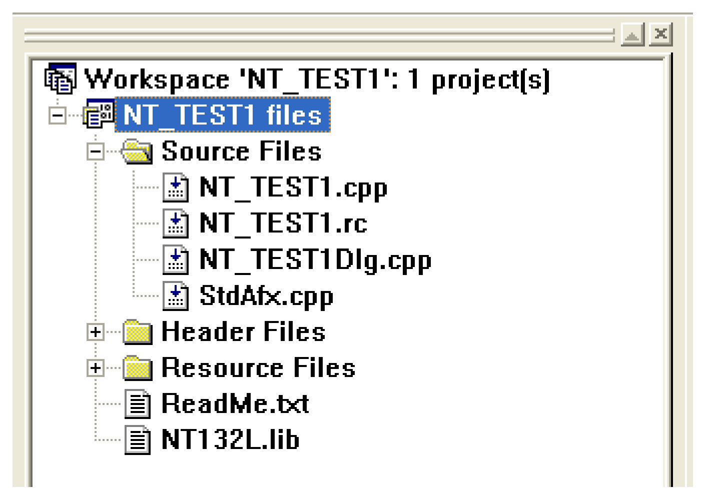


图 5.7 添加 NT132L.LIB 文件后的 FileView 页面

(4) 在FileView页面双击打开“NT_TEST1Dlg.cpp”文件，然后在此文件的开头处包含NTDefine.h和NT132.h文件，如程序清单5.1所示。

程序清单 5.1 增加 NT132 的头文件包含

```
// NT_TEST1Dlg.cpp : implementation file
//

#include "stdafx.h"
#include "NT_TEST1.h"
#include "NT_TEST1Dlg.h"

// 包含NT132的头文件
#include "NTDefine.h"
#include "NT132.h"
......
```

(5) 编辑NT_TEST1对话框的界面，先删除原来有的控件(如OK、Cancel按钮等)，再添加一个按钮(Button)，并将其标题属性改为“查找设备”，如图5.8所示。

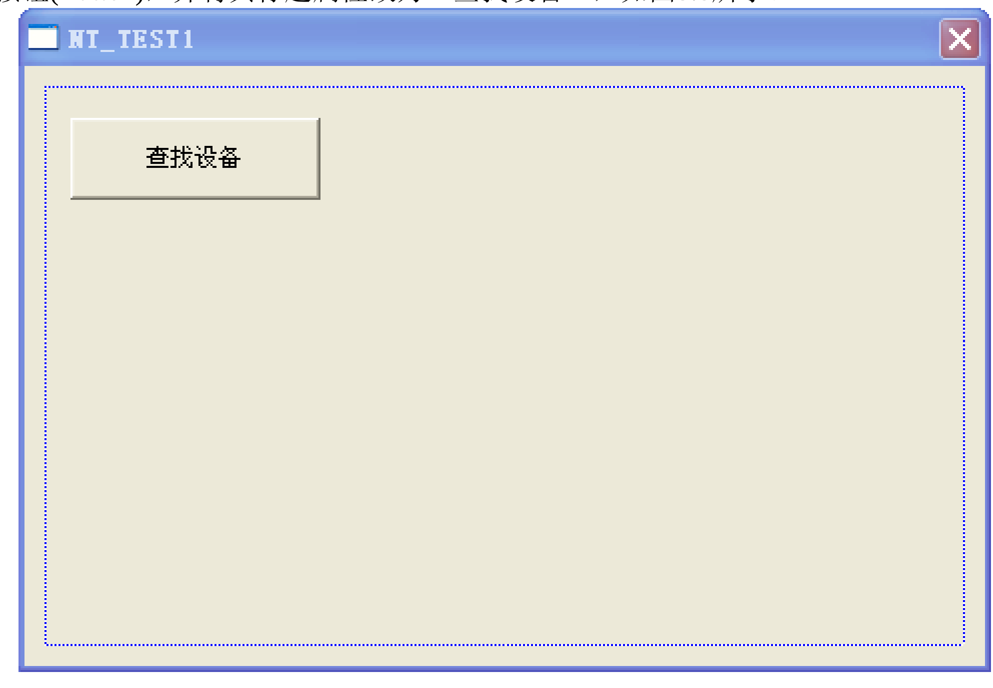


图 5.8　添加“查找设备”按钮

(6) 双击“查找设备”按钮，添加此按钮的单击事件处理代码，如程序清单5.2所示。程序中固定查找产品编号为16个‘0’的加密锁，此产品编号是加密锁出厂默认设置值，如果用户已经产生过新的产品编号(比如使用NT132 Demo来修改)，则需要设置ucPIDBuf为正确的产品编号值。

程序清单 5.2　查找设备的代码

```
void CNT_TEST1Dlg::OnFindNTLock()
{
    // TODO: Add your control notification handler code here
    unsigned char ucPIDBuf[16];    // 产品编号缓冲区
    int     iDevNo;
    int     i;

    // 设置产品编号为16个'0' (请根据实际情况修改)
    for(i=0; i<16; i++)
    {
        ucPIDBuf[i] = '0';
    }

    // 查找产品编号为"0000000000000000"的加密锁
    iDevNo = NT132FindDev(ucPIDBuf);
    if(0 == iDevNo)
    {
        AfxMessageBox("没有找到此产品编号的设备!");
    }
    else
    {
        AfxMessageBox("已找到设备!");
    }
}
```

编写好以上代码后，就可以编译运行程序了，首先将加密锁插入PC机的USB接口，然后单击“查找设备”按钮，看看程序运行结果。后面的步骤也一样，每添加完一个功能按钮及其处理代码后，即可编译运行来看结果。

(7) 添加一个按钮(Button)，并将其标题属性改为“打开设备”。在NT_TEST1Dlg.cpp文件开头处添加一个全局变量GhdFDLock，以用来保存(打开的)加密锁的设备句柄，如程序清单5.3所示。

然后双击“打开设备”按钮，添加此按钮的单击事件处理代码，如程序清单5.4所示。程序中固定打开产品编号为16个‘0’的加密锁，此产品编号是加密锁出厂默认设置值，如果用户已经产生过新的产品编号(比如使用NT132Edit工具软件来修改)，则需要设置ucPIDBuf为正确的产品编号值。

程序清单 5.3　定义 GhdFDLock 变量

```
// NT_TEST1Dlg.cpp : implementation file
//

#include "stdafx.h"
#include "NT_TEST1.h"
#include "NT_TEST1Dlg.h"

// 包含NT132的头文件
#include "NTDefine.h"
#include "NT132.h"


// 定义全局变量
HANDLE  GhdFDLock = INVALID_HANDLE_VALUE;
......
```

程序清单 5.4　打开设备的代码

```
void CNT_TEST1Dlg::OnOpenNTLock()
{
   // TODO: Add your control notification handler code here
   unsigned char ucPIDBuf[16];    // 产品编号缓冲区
   int    i;

   // 设置产品编号为16个'0'  (请根据实际情况修改)
   for(i=0; i<16; i++)
   {
      ucPIDBuf[i] = '0';
   }

   // 打开产品编号为"0000000000000000"的加密锁
   GhdFDLock = NT132OpenDev(ucPIDBuf, 0);

   if((int)GhdFDLock > 0)
   {
      AfxMessageBox("设备已打开.");
   }
   else
   {
      AfxMessageBox("打开设备失败!");
   }
}
```

(8) 添加两个按钮，将它们的标题属性改为“点亮指示灯”和“熄灭指示灯”，然后为“点亮指示灯”按钮添加单击事件处理代码，如程序清单5.5所示的OnLEDOn()函数。再为“熄灭指示灯”按钮添加单击事件处理代码，如程序清单5.5所示OnLEDOff()函数。

程序清单 5.5 指示灯控制代码

```
void CNT_TEST1Dlg::OnLEDOn()
{
    // TODO: Add your control notification handler code here
    int  iOpStat;               // 操作状态变量

    if((int)GhdFDLock > 0)      // 判断设备是否已打开
    {
        iOpStat = NT132LedOpen(GhdFDLock);      // 控制指示灯点亮

        if(iOpStat != OP_OK)
        {
            AfxMessageBox("操作失败!");
        }
    }
    else
    {
        AfxMessageBox("请先打开设备!");
    }
}


void CNT_TEST1Dlg::OnLEDOff()
{
    // TODO: Add your control notification handler code here
    int  iOpStat;               // 操作状态变量

    if((int)GhdFDLock > 0)      // 判断设备是否已打开
    {
        iOpStat = NT132LedClose(GhdFDLock);     // 控制指示灯熄灭

        if(iOpStat != OP_OK)
        {
            AfxMessageBox("操作失败!");
        }
    }
    else
    {
        AfxMessageBox("请先打开设备!");
    }
}
```

(9) 添加一个按钮，将它的标题属性改为“获取设备编号(DID)”，然后为它添加单击事件处理代码，如程序清单5.6所示。

程序清单 5.6 获取设备编号

```
void CNT_TEST1Dlg::OnGetDID()
{
    // TODO: Add your control notification handler code here
    char  cDIDBuf[13];       // DID缓冲区, 12字节DID字符, 1字节结束符号'\0'
```

```
    int   iOpStat;          // 操作状态变量

    if((int)GhdFDLock > 0)  // 判断设备是否已打开
    {
        // 获取DID (12字节字符)
        iOpStat = NT132GetDID(GhdFDLock, (unsigned char *)cDIDBuf);
        cDIDBuf[12] = '\0'; // 设置字符串结束符

        if(iOpStat == OP_OK)
        {
            AfxMessageBox(cDIDBuf);// 显示获取到的DID
        }
        else
        {
            AfxMessageBox("操作失败!");
        }
    }
    else
    {
        AfxMessageBox("请先打开设备!");
    }
}
```

(10) 添加一个按钮，将它的标题属性改为“选择操作扇区”，然后为它添加单击事件处理代码，如程序清单5.7所示。程序中直接调用NT132SelectSectors接口函数来选择扇区0，然后根据其返回值判断初始化是否成功。

程序清单 5.7　选择操作扇区的代码

```
void CNT_TEST1Dlg::OnSelectSector()
{
    // TODO: Add your control notification handler code here
    int   iOpStat;          // 操作状态变量


    if((int)GhdFDLock > 0)  // 判断设备是否已打开
    {
        iOpStat = NT132SelectSectors(GhdFDLock, 0);    // 选择扇区0

        if(iOpStat == OP_OK)
        {
            AfxMessageBox("选择扇区成功!");
        }
        else
        {
            AfxMessageBox("操作失败!");
        }
    }
    else
    {
        AfxMessageBox("请先打开设备!");
    }
}
```

(11) 添加一个按钮，将它的标题属性改为“校验普通用户密码”，然后为它添加单击事件处理代码，如程序清单5.8所示。程序中使用了16个‘8’为密码去校验(校验当前选择扇区

的普通用户密码)，这是加密锁出厂默认的普通用户密码，如果用户修改过此密码(比如使用NT132 Demo来修改)，则需要设置ucUserPassWordBuf为正确的密码值。

程序清单 5.8 校验普通用户密码的代码

```
void CNT_TEST1Dlg::OnVerifyUserPass()
{
    // TODO: Add your control notification handler code here
    unsigned char  ucUserPassWordBuf[16]; // 16字节普通用户密码缓冲区
    int    iVerifyNum;                     // 密码校验次数
    int    i;

    // 判断设备是否已打开
    if((int)GhdFDLock <= 0)
    {
        AfxMessageBox("请先打开设备!");
        return;
    }

    // 设置密码为16个'8', 即使用16个'8'为密码去校验 (请根据实际情况修改)
    for(i=0; i<16; i++)
    {
        ucUserPassWordBuf[i] = '8';
    }

    // 校验当前选择扇区的普通用户密码 (16字节密码)
    iVerifyNum = NT132VerifyUserPassword(GhdFDLock,
                                          ucUserPassWordBuf, 16);

    if(iVerifyNum > 0)
    {
        AfxMessageBox("校验成功, 已切换到普通用户权限.");
    }
    else
    {
        AfxMessageBox("校验失败, 已切换到匿名权限!");
    }
}
```

(12) 添加一个按钮，将它的标题属性改为“读取存储区地址0--9的数据”，然后为它添加单击事件处理代码，如程序清单5.9所示。程序先从用户数据存储区的0地址开始读连续10字节数据，读到数据后再以16进制形式显示。

程序清单 5.9 读取用户数据存储区的数据

```
void CNT_TEST1Dlg::OnReadEEPROM()
{
    // TODO: Add your control notification handler code here
    unsigned char  ucDataBuf[10];  // 数据缓冲区
    int   iOpStat;                 // 操作状态变量

    CString  cstrDisp;


    // 判断设备是否已打开
    if((int)GhdFDLock <= 0)
    {
```

```
        AfxMessageBox("请先打开设备!");
        return;
    }

    // 读取当前扇区的用户数据存储区地址0--9的数据
    iOpStat = NT132DevRead(GhdFDLock, 0, ucDataBuf, 10);
    if(iOpStat != OP_OK)
    {
        AfxMessageBox("操作失败!");
        return;
    }

    // 以16进制形式显示读到的数据
    cstrDisp.Format("数据: %02X %02X %02X %02X %02X %02X %02X %02X %02X %02X. ",
        ucDataBuf[0], ucDataBuf[1], ucDataBuf[2], ucDataBuf[3], ucDataBuf[4],
        ucDataBuf[5], ucDataBuf[6], ucDataBuf[7], ucDataBuf[8], ucDataBuf[9] );
    AfxMessageBox(cstrDisp);
}
```

(13) 添加一个按钮，将它的标题属性改为“修改存储区地址0--9的数据”，然后为它添加单击事件处理代码，如程序清单5.10所示。程序先从用户数据存储区的0地址开始读连续10字节数据，然后对第一字节数据进行加1处理，最后再写回用户数据存储区中。

程序清单 5.10 修改用户数据存储区的数据

```
void CNT_TEST1Dlg::OnWriteEEPROM()
{
    // TODO: Add your control notification handler code here
    unsigned char  ucDataBuf[10]; // 数据缓冲区
    int    iOpStat;                // 操作状态变量
    int    i;

    // 判断设备是否已打开
    if((int)GhdFDLock <= 0)
    {
        AfxMessageBox("请先打开设备!");
        return;
    }

    // 读取当前扇区的用户数据存储区地址0--9的数据
    iOpStat = NT132DevRead(GhdFDLock, 0, ucDataBuf, 10);
    if(iOpStat != OP_OK)
    {
        AfxMessageBox("操作失败!");
        return;
    }

    // 修改数据
    for(i=0; i<10; i++)
    {
        ucDataBuf[i] = ucDataBuf[i] + 1;
    }

    // 将数据写回当前扇区的用户数据存储区
    iOpStat = NT132DevWrite(GhdFDLock, 0, ucDataBuf, 10);
```

```
    if(iOpStat == OP_OK)
    {
        AfxMessageBox("修改成功.");
    }
    else
    {
        AfxMessageBox("操作失败!");
    }
}
```

编译动行程序后，先查找和打开设备，接着校验普通用户密码，再单击一次“读取存储区地址0--9的数据”按钮，记住读出的数据是多少。然后单击一次“修改存储区地址0--9的数据”按钮，再单击一次“读取存储区地址0--9的数据”按钮，观察数据是否已经被修改了。

(14) 添加一个按钮，将它的标题属性改为“关闭设备”，然后为它添加单击事件处理代码，如程序清单5.11所示。

程序清单 5.11 关闭设备的代码

```
void CNT_TEST1Dlg::OnCloseNTLock()
{
    // TODO: Add your control notification handler code here
    int   iOpStat;                    // 操作状态变量

    // 判断设备是否已打开
    if((int)GhdFDLock > 0)
    {
        iOpStat = NT132CloseDev(GhdFDLock);
        GhdFDLock = INVALID_HANDLE_VALUE;
        if(iOpStat == OP_OK)
        {
            AfxMessageBox("设备关闭成功.");
        }
        else
        {
            AfxMessageBox("操作失败!");
        }
    }
    else
    {
        AfxMessageBox("设备没有打开!");
        return;
    }
}
```

(15) 至此，这个例子的功能已设计完成，最终的界面如图5.9所示。

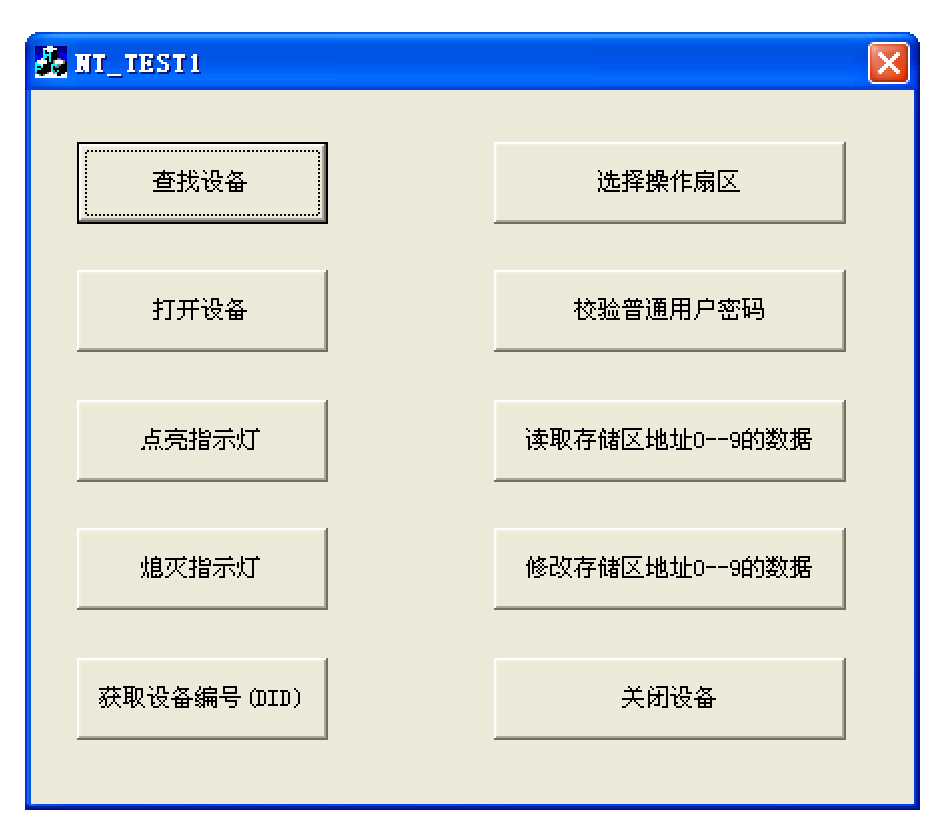


图 5.9 NT_TEST1 操作界面

在这个例子中，并没有去校验超级用户密码(即没有切换到超级用户权限)，因为在进行软件加密时，一般都不会去校验超级用户密码，这样超级用户密码就牢牢地掌握在软件开发商的手上。一般来说，超级用户密码只是软件开发商在初始化加密锁时使用。

5.3.2 基于 DLL 动态库方式

(1) 启动Visual C++ 6.0，新建立一个基于对话框的工程NT_TEST2，工程保存路径为D:\，参考“基于LIB静态库方式”例子的第1步。

(2) 将产品配套光盘“\DLL”目录下的NT132.lib、NT132.dll、NT132.H、NTDefine.H等四个文件复制到D:\NT_TEST2目录下。**注意，复制的是产品配套光盘“\DLL”目录下的文件。**

NT132.lib是静态链接库文件，它包含了NT132.dll导出的符号名和可选的符号，但并不包含实际的代码，所以NT132.dll文件必须和编译生成的应用程序(EXE文件)放在同一个目录下，否则应用程序不能正常运行。

NT132.lib文件不包含有实际的代码，所以其文件大小也比“基于LIB静态库方式”例子中的NT132L.lib文件要小得多。

(3) 参考“基于LIB静态库方式”例子的第3步，将工程目录下的NT132.LIB文件添加进工程。

(4) 在FileView页面双击打开“NT_TEST2Dlg.cpp”文件，然后在此文件的开头处包含NTDefine.h和NT132.h文件。

(5) 按照“基于LIB静态库方式”例子的第5～13步，定义全局变量GhdFDLock，添加所有按钮及相应的代码。

(6) 编译并运行程序，测试各项功能是否正常。

6 API 函数说明

NT 系列加密锁提供有多种 API 接口调用方式(参见第 5 章的说明)，但不管使用哪一种方式，其 API 接口函数都是相同的(函数名、入口参数、返回值等等)。

6.1 操作状态码定义

以下是调用各 API 函数返回的操作状态码(即返回值)及其说明。

- OP_OK 0 表示操作成功；
- MODE_ERR 1 表示当前操作权限不够；
- EEPROM_ERR 2 表示写数据失败；
- PIN_ERR 3 表示校验密码失败；
- CHK_ZERO 4 表示没有校验次数(密码已被锁死)；
- PARA_ERR 5 表示输入的参数不正确，请检查输入参数；
- CMD_ERR 6 表示输入的命令错误，或当前设备不支持该命令；
- SEND_ERR -1 表示发送数据失败；
- REC_ERR -2 表示接收数据失败；
- EN_ERR -3 表示写数据错误，请检查输入数据是否正确；
- DE_ERR -4 表示读数据错误，请重新操作一次；
- READ_ERR -5 表示读数据失败；
- OPEN_ERR -6 表示设备已打开，请不要再打开设备；
- OPENNOT_ERR -7 表示设备没有打开，请重新打开设备；
- PASS_ERR -8 表示校验密码错误。

6.2 查找设备

查找设备的API函数为NT132FindDev，函数详细描述如 表 6.1 所示。

表 6.1 NT132FindDev 函数

函数原型	short __stdcall NT132FindDev(const unsigned char *pPidData);
功能	查找指定产品编号(PID)的在线设备个数
输入参数	pPidData 16 字节产品编号(PID)的缓冲区指针
输出参数	无
返回值	查找到的设备个数（>0 表示找到相应的设备)
权限类别	匿名

使用示例	unsigned char ucPidDataBuf[16];
	CString cstrPID;
	int iDevNo;
	
	cstrPID = "0000000000000000";
	memcpy(ucPidDataBuf, cstrPID, sizeof(ucPidDataBuf));
	iDevNo = NT132FindDev(ucPidDataBuf);
	if(iDevNo == 0)
	{
	    AfxMessageBox("没有找到此产品编号的设备!", MB_ICONINFORMATION);
	    return;
	}

6.3 打开设备

打开设备的API函数为NT132OpenDev，函数详细描述如 表 6.2 所示。

表 6.2 NT132OpenDev 函数

函数原型	HANDLE __stdcall NT132OpenDev(const unsigned char *pPidData, short Index);
功能	打开指定产品编号(PID)及索引的设备
输入参数	pPidData 16 字节产品编号的缓冲区指针 Index 要打开的设备索引(从 0 开始，比如同时使用 3 个加密锁，则它们的设备索引分别为 0、1、2)
输出参数	无
返回值	相应的设备操作句柄 (>0 为正确)
权限类别	匿名

使用示例	HANDLE GhdFDLock = INVALID_HANDLE_VALUE;
	……
	unsigned char ucPidDataBuf[16];
	CString cstrPID;
	
	cstrPID = "0000000000000000";
	memcpy(ucPidDataBuf, cstrPID, sizeof(ucPidDataBuf));
	GhdFDLock = NT132OpenDev(ucPidDataBuf, 0);
	if((int) GhdFDLock < 1)
	{
	AfxMessageBox("打开设备失败!", MB_ICONINFORMATION);
	return;
	}

6.4 指示灯点亮

控制指示灯点亮的API函数为NT132LedOpen，函数详细描述如 表 6.3 所示。

表 6.3 NT132LedOpen 函数

函数原型	short __stdcall NT132LedOpen(HANDLE handle);
功能	控制当前设备的指示灯点亮
输入参数	handle 当前设备的操作句柄
输出参数	无
返回值	返回操作状态 (OP_OK 为正确)
权限类别	匿名

使用示例	short sOpStat;
	
	sOpStat = NT132LedOpen(GhdFDLock);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("指示灯已点亮!", MB_ICONINFORMATION);
	return;
	}

6.5 指示灯熄灭

控制指示灯熄灭的API函数为NT132LedClose，函数详细描述如 表 6.4 所示。

表 6.4 NT132LedClose 函数

函数原型	short __stdcall NT132LedClose(HANDLE handle);
功能	控制当前设备的指示灯熄灭
输入参数	handle 当前设备的操作句柄
输出参数	无
返回值	返回操作状态 (OP_OK 为正确)
权限类别	匿名
使用示例	short sOpStat; sOpStat = NT132LedClose(GhdFDLock);

	if(sOpStat == OP_OK)
	{
	AfxMessageBox("指示灯已熄灭!", MB_ICONINFORMATION);
	return;
	}

6.6 获取设备编号

获取设备编号(DID)的API函数为NT132GetDID，函数详细描述如 表 6.5 所示。

表 6.5 NT132GetDID 函数

函数原型	short __stdcall NT132GetDID(HANDLE handle, unsigned char *pDidData);
功能	获取设备编号
输入参数	handle 当前设备的操作句柄
输出参数	pDidData 用来保存 12 字节设备编号的缓冲区指针
返回值	返回操作状态 (OP_OK 为正确)
权限类别	匿名

使用示例	unsigned char ucDidDataBuf[12];
	short sOpStat;
	
	sOpStat = NT132GetDID(GhdFDLock, ucDidDataBuf);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("获取设备编号成功!", MB_ICONINFORMATION);
	return;
	}

6.7 获取产品编号

获取产品编号(PID)的API函数为NT132GetPID，函数详细描述如 表 6.6 所示。

表 6.6 NT132GetPID 函数

函数原型	short __stdcall NT132GetPID(HANDLE handle, unsigned char *pPidData);
功能	获取产品编号
输入参数	handle 当前设备的操作句柄
输出参数	pPidData 用来保存 16 字节产品编号的缓冲区指针
返回值	返回操作状态 (OP_OK 为正确)
权限类别	匿名

使用示例	unsigned char ucPidDataBuf[16];
	short sOpStat;
	
	sOpStat = NT132GetPID(GhdFDLock, ucPidDataBuf);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("获取产品编号成功!", MB_ICONINFORMATION);
	return;
	}

6.8 选择操作扇区

选择操作扇区的API函数为NT132SelectSectors，函数详细描述如 表 6.7 所示。

表 6.7 NT132SelectSectors 函数

函数原型	short __stdcall NT132SelectSectors(HANDLE handle, short SectorsID);
功能	选择要操作的扇区
输入参数	handle 当前设备的操作句柄

输出参数	SectorsID 选择的扇区号，值为 0～3
返回值	返回操作状态 (OP_OK 为正确)
权限类别	匿名

使用示例	short sOpStat;
	
	sOpStat = NT132SelectSectors(GhdFDLock, 0); // 选择扇区 0
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("选择扇区操作成功!", MB_ICONINFORMATION);
	return;
	}

6.9 校验超级用户密码

校验超级用户密码的API函数为NT132VerifySuperPassword，函数详细描述如 表 6.8 所示。

表 6.8 NT132VerifySuperPassword 函数

函数原型	short __stdcall NT132VerifySuperPassword(HANDLE handle, const unsigned char *pSuperPassword, short nLen);
功能	校验超级用户密码，如果校验成功则把当前操作权限提升为超级用户权限，如果校验失败则返回到匿名权限
输入参数	handle 当前设备的操作句柄 pSuperPassword 密码缓冲区指针(1～16 字节密码) nLen 密码长度(应为 1～16)
输出参数	无
返回值	密码的校验次数 (>0 正确)
权限类别	匿名

使用示例	unsigned char ucSuperPassWordBuf[16];
	CString cstrVerifyPass;
	short sVerifyNum;
	
	cstrVerifyPass = "8888888888888888";
	memcpy(ucSuperPassWordBuf, cstrVerifyPass, cstrVerifyPass.GetLength());
	sVerifyNum = NT132VerifySuperPassword (GhdFDLock, ucSuperPassWordBuf,
	cstrVerifyPass.GetLength());
	if(sVerifyNum == 0)
	{
	AfxMessageBox("对不起，密码校验次数为0!", MB_ICONINFORMATION);
	return;
	}
	
	if(sVerifyNum < 0)
	{
	AfxMessageBox("对不起，密码校验失败!", MB_ICONINFORMATION);
	return;
	}
	
	AfxMessageBox("校验密码成功!", MB_OK|MB_ICONINFORMATION);

6.10 校验普通用户密码

校验普通用户密码的API函数为NT132VerifyUserPassword，函数详细描述如 表 6.9 所示。

表 6.9 NT132VerifyUserPassword 函数

函数原型	short __stdcall NT132VerifyUserPassword(HANDLE handle, const unsigned char *pUserPassword, short nLen);
功能	校验**已选择扇区的**普通用户密码，如果校验成功则把当前操作权限提升为普通用户权限，如果校验失败则返回到匿名权限。 调用本函数之前，先要调用 NT132SelectSectors()函数来选择待操作扇区。

输入参数	handle 当前设备的操作句柄 pUserPassword 密码缓冲区指针(1～16 字节密码) nLen 密码长度(应为 1～16)
输出参数	无
返回值	密码的校验次数（>0 正确)
权限类别	匿名

使用示例	unsigned char ucUserPassWordBuf[16];
	CString cstrVerifyPass;
	short sVerifyNum;
	short sOpStat;
	
	sOpStat = NT132SelectSectors(GhdFDLock, 2); // 选择扇区 2
	if(sOpStat != OP_OK)
	{
	AfxMessageBox("选择扇区操作失败!", MB_ICONINFORMATION);
	return;
	}
	
	cstrVerifyPass = "8888888888888888";
	memcpy(ucUserPassWordBuf, cstrVerifyPass, cstrVerifyPass.GetLength());
	sVerifyNum = NT132VerifyUserPassword(GhdFDLock, ucUserPassWordBuf,
	cstrVerifyPass.GetLength());
	if(sVerifyNum == 0)
	{
	AfxMessageBox("对不起，密码校验次数为0!", MB_ICONINFORMATION);
	return;
	}
	
	if(sVerifyNum < 0)
	{
	AfxMessageBox("对不起，密码校验失败!", MB_ICONINFORMATION);
	return;
	}
	
	AfxMessageBox("校验密码成功!", MB_OK|MB_ICONINFORMATION);

6.11 修改超级用户密码

修改超级用户密码的API函数为NT132SetSuperPassword，函数详细描述如 表 6.10 所示。

表 6.10 NT132SetSuperPassword 函数

函数原型	short __stdcall NT132SetSuperPassword(HANDLE handle, const unsigned char *pSuperPassword, short nLen);
功能	修改超级用户密码
输入参数	handle 当前设备的操作句柄 pSuperPassword 密码缓冲区指针(1～16 字节密码) nLen 密码长度(应为 1～16)
输出参数	无
返回值	返回操作状态 (OP_OK 为正确)
权限类别	超级用户权限

使用示例	unsigned char ucSuperPassWordBuf[16];
	CString cstrNewPass;
	short sOpStat;
	
	cstrNewPass ="ok";
	memcpy(ucSuperPassWordBuf, cstrNewPass, cstrNewPass.GetLength());
	sOpStat = NT132SetSuperPassword(GhdFDLock, ucSuperPassWordBuf,
	cstrNewPass.GetLength());
	if(sOpStat == OP_OK)
	{

	AfxMessageBox("修改密码成功!", MB_ICONINFORMATION); return; }

6.12　修改普通用户密码

修改普通用户密码的API函数为NT132SetUserPassword，函数详细描述如 表 6.11 所示。

表 6.11　NT132SetUserPassword 函数

函数原型	short __stdcall NT132SetUserPassword(HANDLE handle, const unsigned char *pUserPassword, short nLen);
功能	修改当前扇区的普通用户密码
输入参数	handle　当前设备的操作句柄 pUserPassword　密码缓冲区指针(1～16 字节密码) nLen　密码长度(应为 1～16)
输出参数	无
返回值	返回操作状态 (OP_OK 为正确)
权限类别	普通用户或超级用户权限

使用示例	unsigned char ucUserPassWordBuf[16];
	CString cstrNewPass;
	short sOpStat;
	
	cstrNewPass ="ok";
	memcpy(ucUserPassWordBuf, cstrNewPass, cstrNewPass.GetLength());
	sOpStat = NT132SetUserPassword(GhdFDLock, ucUserPassWordBuf,
	cstrNewPass.GetLength());
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("修改密码成功!", MB_ICONINFORMATION);
	return;
	}

6.13　设置密码校验次数

设置密码校验次数的API函数为NT132SetVerifyNum，函数详细描述如 表 6.12 所示。

表 6.12　NT132SetVerifyNum 函数

函数原型	short __stdcall NT132SetVerifyNum(HANDLE handle, short SuperNum, short UserNum0, short UserNum1, short UserNum2, short UserNum3);
功能	设置超级用户、各扇区的普通用户密码校验次数
输入参数	handle　当前设备的操作句柄 SuperNum　超级用户密码校验次数 (0 为不限次数) UserNum0　扇区 0 的普通用户密码校验次数(0 为不限) UserNum1　扇区 1 的普通用户密码校验次数(0 为不限) UserNum2　扇区 2 的普通用户密码校验次数(0 为不限) UserNum3　扇区 3 的普通用户密码校验次数(0 为不限)
输出参数	无
返回值	返回操作状态 (OP_OK 为正确)
权限类别	超级用户权限

使用示例	short sOpStat;
	
	sOpStat = NT132SetVerifyNum(GhdFDLock, 0, 3, 5, 8, 5);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("设置密码校验次数成功!", MB_ICONINFORMATION);
	return;
	}

6.14 设置产品显示信息

设置产品显示信息的API函数为NT132SetDispInfo，函数详细描述如 表 6.13 所示。

表 6.13 NT132SetDispInfo 函数

函数原型	short __stdcall NT132SetDispInfo(HANDLE handle, const unsigned char *pDispData, short nLen);
功能	将产品显示信息写到加锁中 (在加密锁属性页显示的信息)
输入参数	handle 当前设备的操作句柄 pDispData 显示信息数据缓冲区指针(17 个汉字或 ASCII 码字符，因为一个汉字占 2 字节空间，所以缓冲区大小应 ≥34 字节) nLen 显示信息长度
输出参数	无
返回值	返回操作状态 (OP_OK 为正确)
权限类别	超级用户权限

使用示例	unsigned char ucDispDataBuf[34];
	CString cstrDisp;
	short sOpStat;
	
	cstrDisp = "www.FDLock.com";
	memcpy(ucDispDataBuf, cstrDisp, cstrDisp.GetLength());
	sOpStat = NT132SetDispInfo(GhdFDLock, ucDispDataBuf, cstrDisp.GetLength());
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("设置产品显示信息成功!", MB_ICONINFORMATION);
	return;
	}

6.15 设置产品编号

设置产品编号(即产生新的产品编号)的API函数为NT132SetPID，函数详细描述如 表 6.14 所示。

表 6.14 NT132SetPID 函数

函数原型	short __stdcall NT132SetPID(HANDLE handle, const unsigned char *pProductPassword, short nBuffLen, unsigned char *pPidData);
功能	根据输入的产品密码，生成并设置新的产品编号(PID)
输入参数	handle 当前设备的操作句柄 pProductPassword 用于生成产品编号的产品密码缓冲区指针 nLen 产品密码的长度 (1～56 字节)
输出参数	pPidData 用来保存生成的新产品编号的缓冲区指针(产品编号为 16 字节)
返回值	返回操作状态 (OP_OK 为正确)
权限类别	超级用户权限

使用示例	unsigned char ucProductPasswordBuf[56];
	unsigned char ucPidDataBuf[16];
	CString cstrProductPass;
	short sOpStat;
	
	memset(ucProductPasswordBuf, 0, sizeof(ucProductPasswordBuf));
	cstrProductPass = "Hello NT132";
	memcpy(ucProductPasswordBuf, cstrProductPass, cstrProductPass.GetLength());
	sOpStat = NT132SetPID(GhdFDLock, ucProductPasswordBuf, cstrProductPass.GetLength(), ucPidDataBuf);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("设置产品编号成功!", MB_ICONINFORMATION);
	return;
	}

6.16 向设备写数据

向设备写数据，即是把数据写入**当前扇区的**用户数据存储空间，对应的API函数为NT132DevWrite，函数详细描述如 表 6.15 所示。

表 6.15 NT132DevWrite 函数

函数原型	short __stdcall NT132DevWrite(HANDLE handle, short StartAdd, const unsigned char *pWriteBuff, short nBuffLen);
功能	将数据写入**当前扇区的**用户数据存储空间，从指定地址开始写指定个数的数据
输入参数	handle 当前设备的操作句柄 StartAdd 指定的起始地址 pWriteBuff 要写入的数据缓冲区指针 nBuffLen 要写入的数据个数
输出参数	无
返回值	返回操作状态 (OP_OK 为正确)
权限类别	普通用户或超级用户权限

使用示例	unsigned char ucWriteDataBuf[128];
	CString cstrMData;
	short sNo;
	short sOpStat;
	
	cstrMData = "1234567890";
	sNo = cstrMData.GetLength();
	if(sNo > 128) sNo = 128;
	
	memcpy(ucWriteDataBuf, cstrMData, sNo);
	sOpStat = NT132DevWrite(GhdFDLock, 0, ucWriteDataBuf, sNo);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("写数据成功!", MB_ICONINFORMATION);
	return;
	}

6.17 从设备读数据

从设备读数据，即是读取**当前扇区的**用户数据存储空间的数据，对应的API函数为NT132DevRead，函数详细描述如 表 6.16 所示。

表 6.16 NT132DevRead 函数

函数原型	short __stdcall NT132DevRead(HANDLE handle, short StartAdd, unsigned char *pReadBuff, short nBuffLen);
功能	读取**当前扇区的**用户数据存储空间，从指定地址开始读指定个数的数据
输入参数	handle 当前设备的操作句柄 StartAdd 指定的起始地址 nBuffLen 要读取的数据个数
输出参数	pReadBuff 用来保存读取到的数据的缓冲区指针
返回值	返回操作状态 (OP_OK 为正确)
权限类别	普通用户或超级用户权限

使用示例	unsigned char ucReadBuf[128];
	short sOpStat;
	
	memset(ucReadBuf, 0, sizeof(ucReadBuf));
	
	sOpStat = NT132DevRead(GhdFDLock, 0, ucReadBuf, 80);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("读取数据成功!", MB_ICONINFORMATION);
	return;
	}

6.18 MD5 加密运算

MD5 加密算法的API函数为NT132MD5EncryptData，函数详细描述如 表 6.17 所示。

表 6.17 NT132MD5EncryptData 函数

函数原型	short __stdcall NT132MD5EncryptData(HANDLE handle, const unsigned char *SourceData, short nBuffLen, char *ObjData);
功能	非标的 MD5 加密算法
输入参数	handle 当前设备的操作句柄 SourceData 源数据缓冲区指针 nBuffLen 数据长度(1～51)
输出参数	ObjData 目标数据指针 (32 字节缓冲区)
返回值	返回操作状态(OP_OK 为正确)
权限类别	匿名

使用示例	unsigned char ucInParaBuf[60];
	unsigned char ucOutParaBuf[60];
	short sOpStat;
	
	ucInParaBuf[0] = 0x11;
	ucInParaBuf[1] = 0x22;
	ucInParaBuf[2] = 0x33;
	ucInParaBuf[3] = 0x44;
	ucInParaBuf[4] = 0x55;
	ucInParaBuf[5] = 0x66;
	ucInParaBuf[6] = 0x77;
	ucInParaBuf[7] = 0x88;
	
	sOpStat = NT132MD5EncryptData(GhdFDLock, ucInParaBuf, 8, (char *)ucOutParaBuf);
	if(sOpStat == OP_OK)
	{ // 校验返回结果(ucOutParaBuf缓冲区的数据)是否正确
	……
	}
	else
	{ // 进行其它处理
	……
	}

6.19 关闭设备

关闭设备的API函数为NT132CloseDev，函数详细描述如 表 6.18 所示。

表 6.18 NT132CloseDev 函数

函数原型	short __stdcall NT132CloseDev(HANDLE handle);
功能	关闭设备，把用户权限恢复到匿名权限
输入参数	handle 当前设备的操作句柄
输出参数	无
返回值	返回操作状态(OP_OK 为正确)
权限类别	匿名

使用示例	short sOpStat;
	
	sOpStat = NT132CloseDev(GhdFDLock);
	if(sOpStat == OP_OK)
	{
	AfxMessageBox("关闭设备成功!", MB_ICONINFORMATION);
	return;
	}

7　工具软件的使用

7.1　编辑器

“NT132编辑器”是软件开发商用来初始化加密锁的量产工具，下文将“NT132编辑器”软件称为NT132Edit。NT132Edit.exe文件在产品光盘的“\NT132_EDIT”目录下。

双击运行NT132Edit，其界面如图7.1所示，界面上有很多复选框，**如果选中某个复选框，表示此项参数设置要写入到加密锁中。**调试时可以只选中其中一项或几项，但如果是量产，一般全部都要选中。

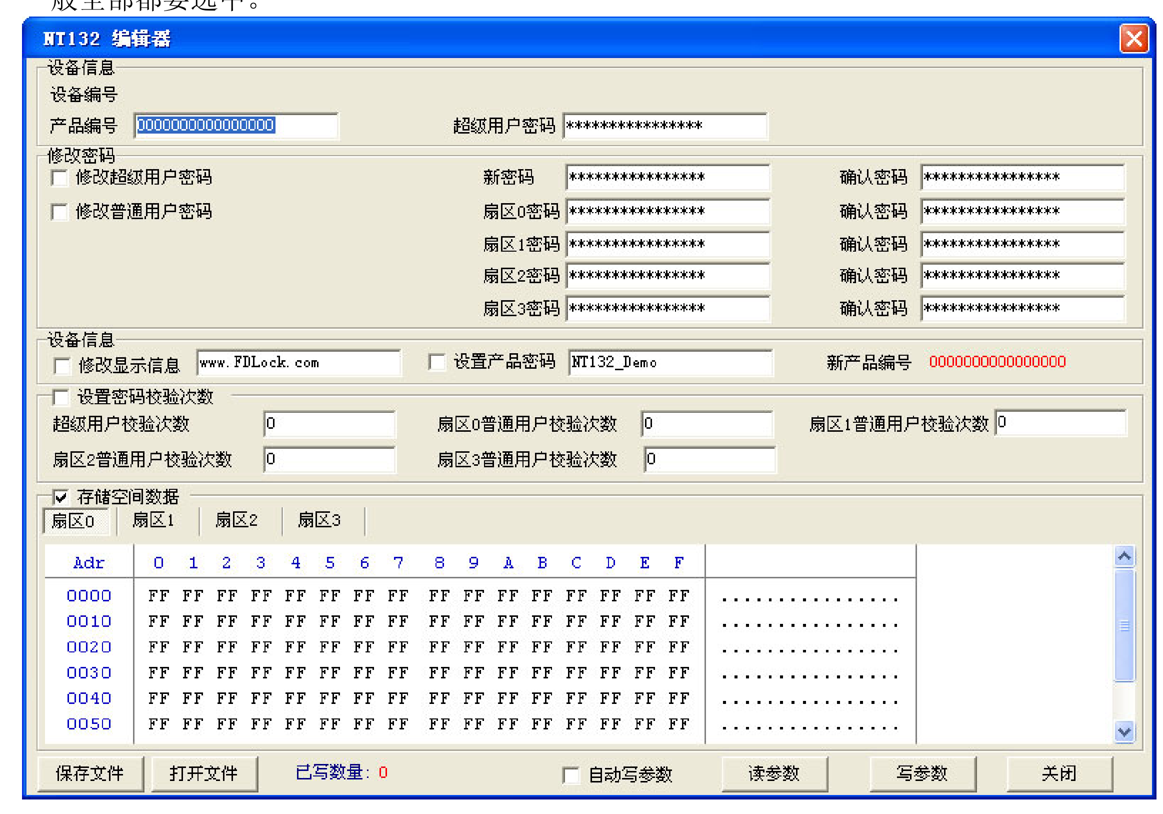


图 7.1　NT132Edit 操作界面

首先，在“产品编号”栏中输入16字节产品编号，例如16个字符‘0’(加密锁的出厂默认设置值)，在“超级用户密码”栏中输入密码，例如16个字符‘8’(加密锁的出厂默认设置值)；

接着，将复选框“修改超级用户密码”选中，并且把要设置的密码输入到它后面的“新密码”和“确认密码”栏中；

将复选框“修改普通用户密码”选中,然后把要设置的各扇区的密码，分别输入到它后面的“扇区x密码(x为0～3)”和“确认密码”栏中；

将复选框“修改显示信息”选中，并在它后面输入产品显示信息(17个汉字或17个字符)；

将复选框“设置产品密码”选中，并在它后面输入用于生成产品编号的产品密码(产品密码可以是1～56个ASCII码字符，也可以是汉字，一个汉字占2个字节)；

将复选框“设置密码校验次数”选中，然后根据自己的需要，在“超级用户校验次数”栏中输入超级用户密码校验次数，在“扇区x普通用户校验次数(x为0～3)”栏中分别输入各扇区的普通用户密码校验次数，值为0～255，其中0表示不限次数；

将复选框“存储空间数据”选中，然后单击“扇区x(x为0～3)”页面，在它下面的16进制编辑区中输入要写到各扇区的用户存储空间的数据，是以16进制形式进行的。编辑区中的左边为存储地址，每一行为16字节数据，右边则是ASCII码显示区(也可在此处直接输入ASCII码)；

设置好各个参数项后，结果参考如图7.2所示。最后，把要初始化的NT132加密锁插入PC的USB接口，单击NT132Edit的“写参数”按钮，即可把各项参数写到加密锁中。另外，还可

以将复选框“自动写参数”选中，此时“写参数”按钮将变为“开始”按钮，单击“开始”按钮，然后每插入一个要初始化的加密锁后，NT132Edit即会自动对它进行初始化操作(即写参数)，初始化完成后PC机会响3声清脆的“De”声。

需要注意的是，单击“写参数”按钮后，加密锁的产品编号和超级用户密码都已经改变了，如果再次点击“写参数”按钮，将会提示“请插入加密狗”(因为没有找到原产品编号的加密锁)。

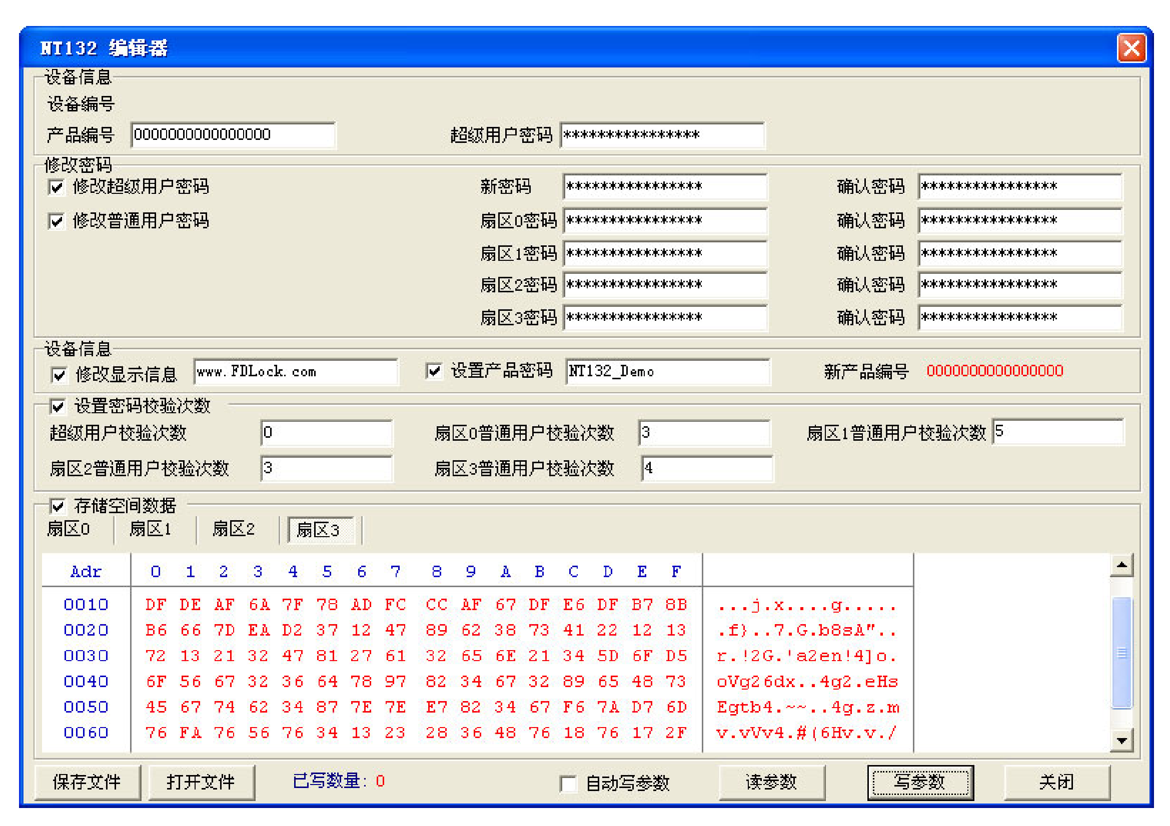


图 7.2 设置好各项参数后的 NT132Edit

另外，还可以把当前的所有设置保存成为一个配置文件，在下次使用时再打开文件调入相应的参数即可，这是通过“保存文件”和“打开文件”按钮来实现的。

NT132Edit中，还有一个“读参数”按钮，使用它只能读出加密锁的设备编号和用户数据存储空间的数据。

7.2 EXE 外壳加密工具

NT132支持EXE外壳加密功能，即直接对可执行文件(*.EXE)进行加密，加密之后，只有插入合法的加密锁，软件才能正确运行。EXE外壳加密需要使用NTShell工具软件来进行，具体操作步骤如下。

1. 将NT132开发套件产品光盘中的“\NT132 EXE文件加密”目录复制到D:\下，参考如图7.3所示。说明：并不一定要复制到D:\下，这里只是为了方便描述才这样做。



图 7.3 复制 NTShell 工具软件

2. 把要加密的EXE文件复制到“D:\NT132 EXE文件加密”目录下，例如图7.4中的calc.exe文件。



图 7.4 把要加密的 EXE 文件放到 NTShell 工作目录下

3. 双击“D:\NT132 EXE文件加密\NTShell.exe”文件，将会弹出如图7.5所示的对话框(NTShell的主界面)。



图 7.5 NTShell 主界面

4. 在 NTShell 的主界面中，有一个“提示标题”和一个“提示内容”栏，这是用来设置弹出的错误对话框的标题和提示信息。也就是说，在运行加密后的 EXE 文件时，如果没有插入合法的加密锁，程序将会停止运行，并弹出一个错误提示对话框，“提示标题”和“提示内容”就是用于这个错误提示对话框的。

用户可根据自己需要，分别修改“提示标题”和“提示内容”栏，参考如图7.6所示。

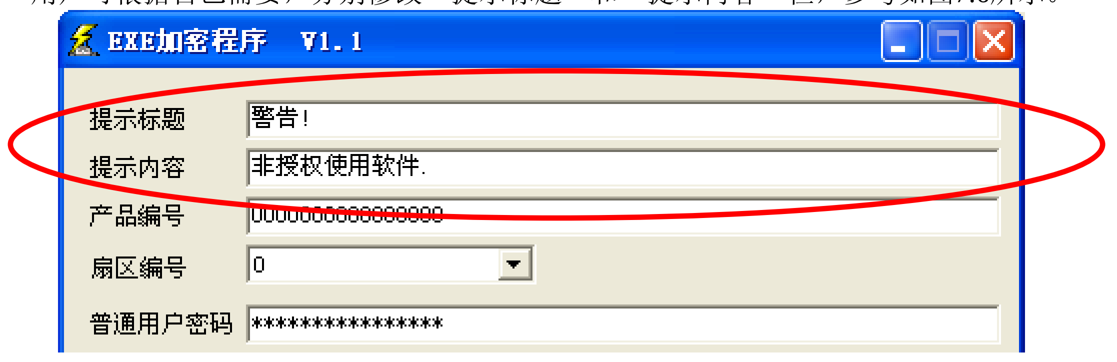


图 7.6 设置错误提示信息

5. 在 NTShell 的主界面中，有一个“产品编号”栏，**这里一定要写入用户最终初始化加密狗时生成的产品编号**。初始化加密狗是使用 NT132Edit 工具软件进行的，具体可参考第 7.1 节的介绍。

如果用户还不知道产品编号是多少，那请打开NT132Edit工具软件，然后插入一个未初始化过的NT132 加密锁；对于外壳加密来说，一般要修改超级用户密码、各扇区的普通用户密码、设置显示信息、设置产品密码即可，在NT132Edit中选中相应的项，并输入自己设置的内容，参考如 图 7.7 所示；最后点击“写参数”，即可初始化加密锁，并得到生成的产品编号，如 图 7.8 所示。**注意，用户一定要记住修改后的超级用户密码、各扇区的普通用户密码，还有新生成的产品编号**。把生成的产品编号写到NTShell中，结果如 图 7.9 所示。

另外，如果用户还没有购买加密锁，只是想试用一下NTShell工具软件，则“产品编码”这一栏可以不用修改。

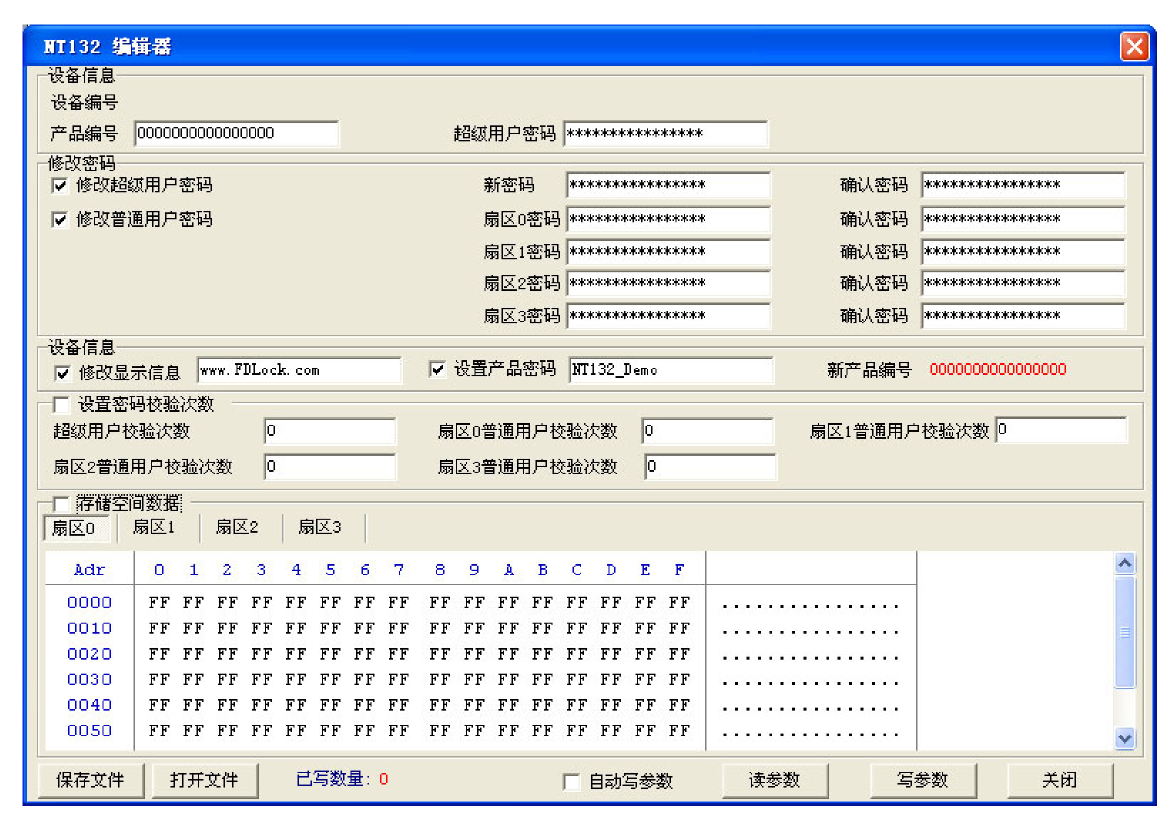


图 7.7 设置加密锁初始化信息

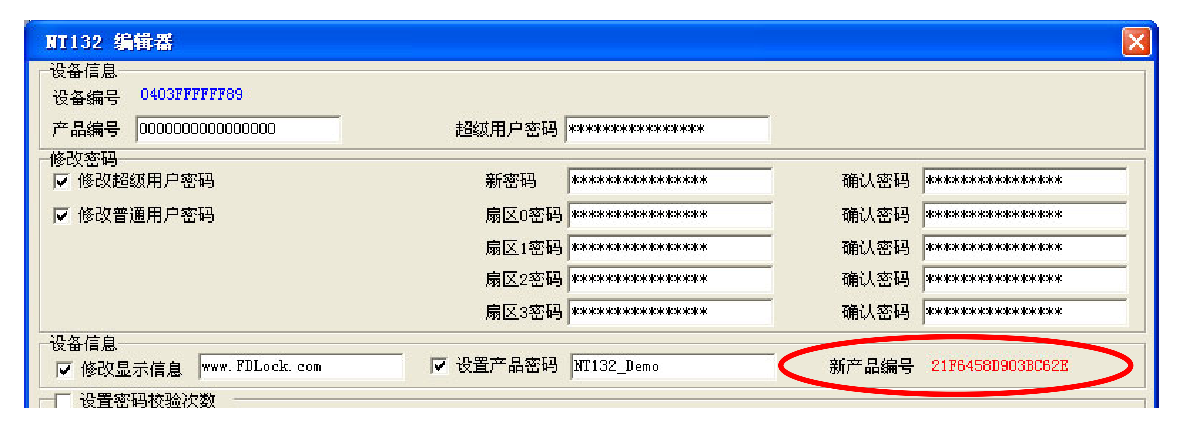


图 7.8 获得产品编号

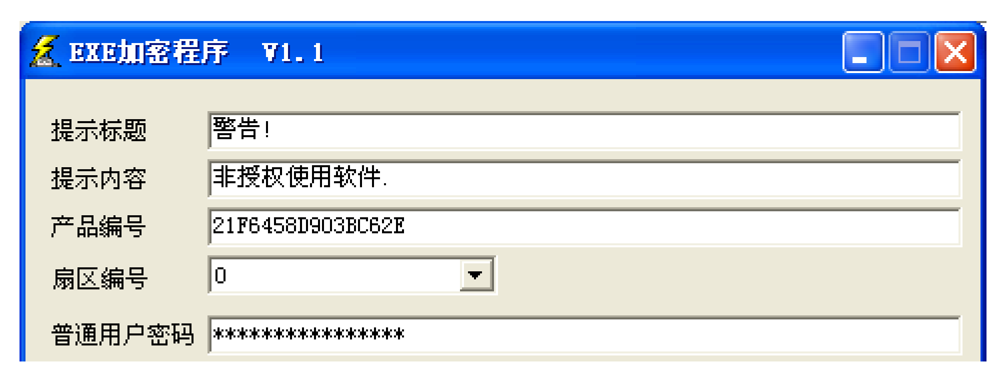


图 7.9 输入正确的产品编号

6. 在 NTShell 的主界面中，在“扇区编号”栏中选择用于加密的扇区，比如扇区 0；接着在“普通用户密码”栏中，**输入所选择扇区的普通用户密码(用户初始化加密狗时修改的)。**

7. 在NTShell的主界面中，单击“选择文件”按钮，在弹出的窗口中操作选择要加密的EXE文件，如“D:\NT132 EXE文件加密\calc.exe”文件，结果如 图 7.10 所示。



图 7.10 选择要加密的 EXE 文件

8. 最后单击“开始加密”按钮，即可对选中的EXE文件进行加密，结果参考如 图 7.11 所示。其中，calc.exe文件是加密后的文件；calc.exe.bak为没加密的EXE文件的备份。



图 7.11 EXE 文件加密结果

9. 双击“D:\NT132 EXE文件加密\calc.exe”，程序会根据 图 7.10 所设置的产品编号和普通用户密码去查找合法的加密锁，如果此时没有插入已经初始化过的合法的加密锁，则会弹出

如 图 7.12 所示的错误提示。



图 7.12　没有加密锁时的错误提示

10. 若插入了合法的加密锁，则calc.exe可以正常运行，如图7.13所示。



图 7.13　calc.exe 正确运行

7.3　DLL 外壳加密工具

对于使用API调试方式来加密软件(参考第5.1节的介绍)，我们推荐使用LIB静态库方式，这样软件的加密强度会更高。但如果用户使用的是Visual Basic、PowerBuilder、FoxPro等开发环境，则只能使用DLL动态库方式。对于使用DLL动态库方式的软件，为了防止DLL劫持，可以使用DllNTShell工具软件来进行DLL外壳加密，具体操作步骤如下。

1. 将NT132开发套件产品光盘中的“\NT132 DLL文件加密”目录复制到D:\下，参考如图7.14所示。说明：并不一定要复制到D:\下，这里只是为了方便描述才这样做。

2. 在进行DLL外壳加密前，用户需要编写和调试好自己的程序。这里以NT132开发套件产品光盘“\example_DLL\VB5”目录下的NT132_DEMO.vbp工程为例进行介绍，首先将产品光盘“\example_DLL\VB5”目录复制到D:\下，然后使用Visual Basic开发环境打开此目录下的NT132_DEMO.vbp工程。

3. 更改程序中对NT132.dll动态库文件的声明或引用，比如原来声明API来源是NT132.dll，现在把它改为MS123.dll(文件名可随意取，但要统一)，参考如图7.15所示。

4. 保存修改好的文件，然后重新编译工程，生成EXE文件(对于NT132_DEMO.vbp工程，编译生成的文件为NT132_DEMO.exe)，然后关闭此工程。



图 7.14 复制 DllNTShell 工具软件

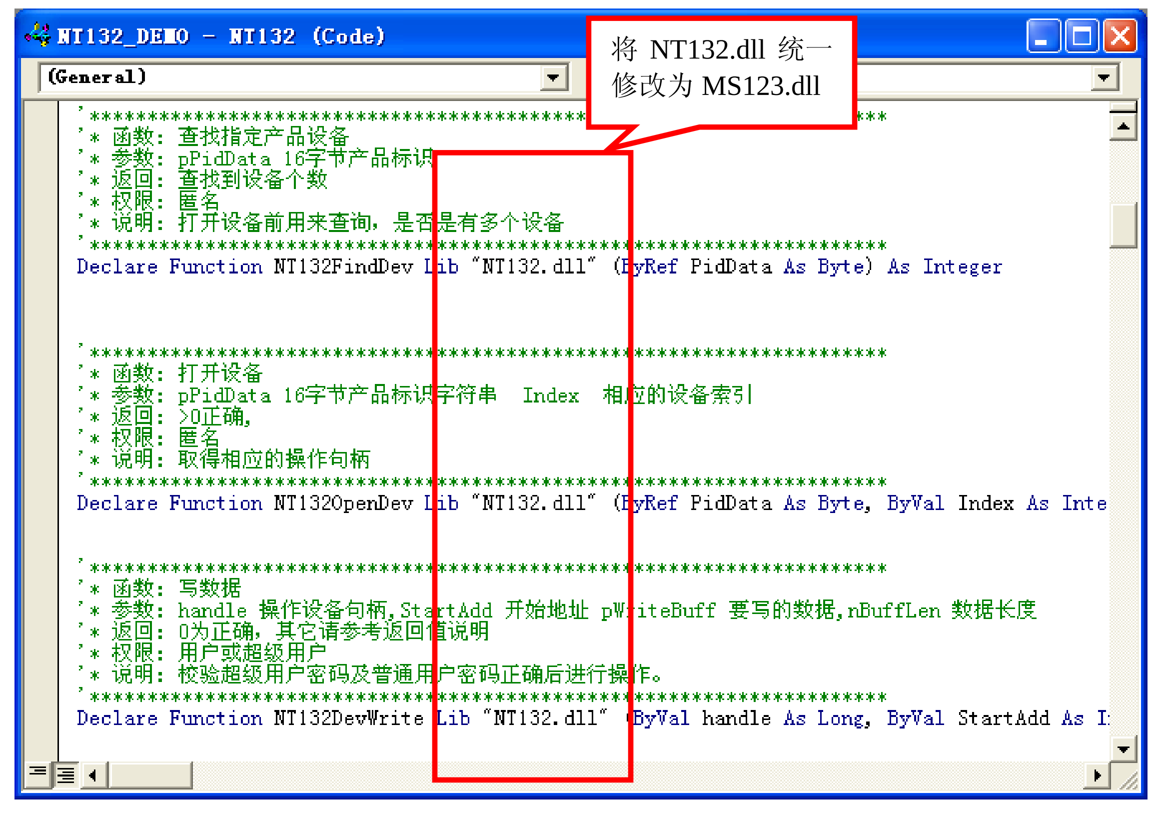


图 7.15 修改对动态库文件的声明或引用

5. 双击“D:\NT132 DLL文件保护\DllNTShell.exe”文件，将会弹出如图7.16所示的对话框(DllNTShell的主界面)。



图 7.16 DllNTShell 主界面

6. 在 DllNTShell 的主界面中，有一个“提示标题”和一个“提示内容”栏，这是用来设置弹出的错误对话框的标题和提示信息。也就是说，在运行加密后的 EXE 文件时，如果没有插入合法的加密锁，程序将会停止运行，并弹出一个错误提示对话框，“提示标题”和“提示内容”就是用于这个错误提示对话框的。

用户可根据自己需要，分别修改“提示标题”和“提示内容”栏，参考如图7.17所示。

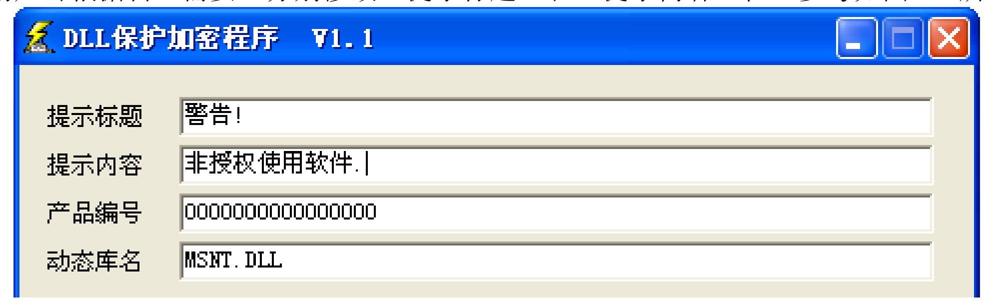


图 7.17 修改错误提示信息

7. 在DllNTShell的主界面中，有一个“产品编号”栏，**这里一定要写入用户最终初始化加密狗时生成的产品编号。**初始化加密狗是使用NT132Edit工具软件进行的，具体可参考第7.1节的介绍，或参考第7.2节第5点的介绍。

另外，如果用户还没有购买加密锁，只是想试用一下DllNTShell工具软件，则“产品编码”这一栏可以不用修改。

8. 在DllNTShell的主界面中，有一个“动态库名”栏，这里一定要输入待加密程序里声明API来源的DLL文件名，比如本小节第3点的MS123.dll，参考如图7.18所示。

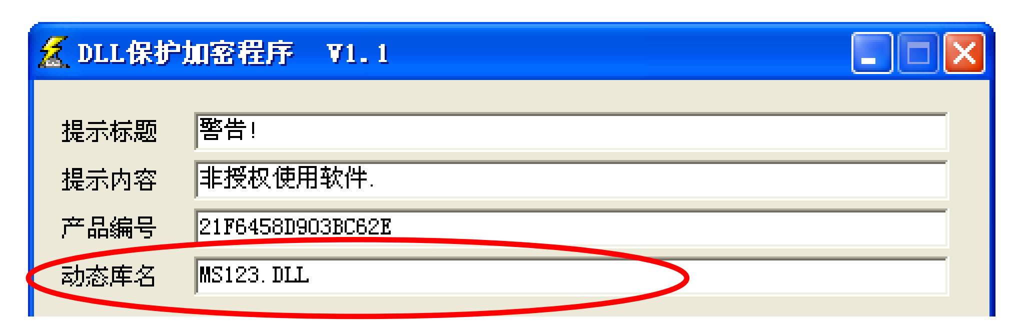


图 7.18 设置程序声明或引用的 DLL 文件名

9. 在DllNTShell的主界面中，单击“选择文件”按钮，在弹出的窗口中操作选择要加密的EXE文件，如“D:\VB5\NT132_DEMO.exe”文件，结果如 图 7.19 所示。

说明：DLL外壳加密并不是直接对DLL文件进行加密，只是将DLL文件注入到需要调用它的EXE文件中，形成一个可独立使用的EXE文件。



图 7.19 选择要 DLL 外壳加密的 EXE 文件

10. 最后单击“开始加密”按钮，即可对选中的EXE文件进行加密，结果参考如图7.20所示。其中，NT132_DEMO.exe文件是加密后的文件；NT132_DEMO.exe.bak为没加密的EXE文件的备份。

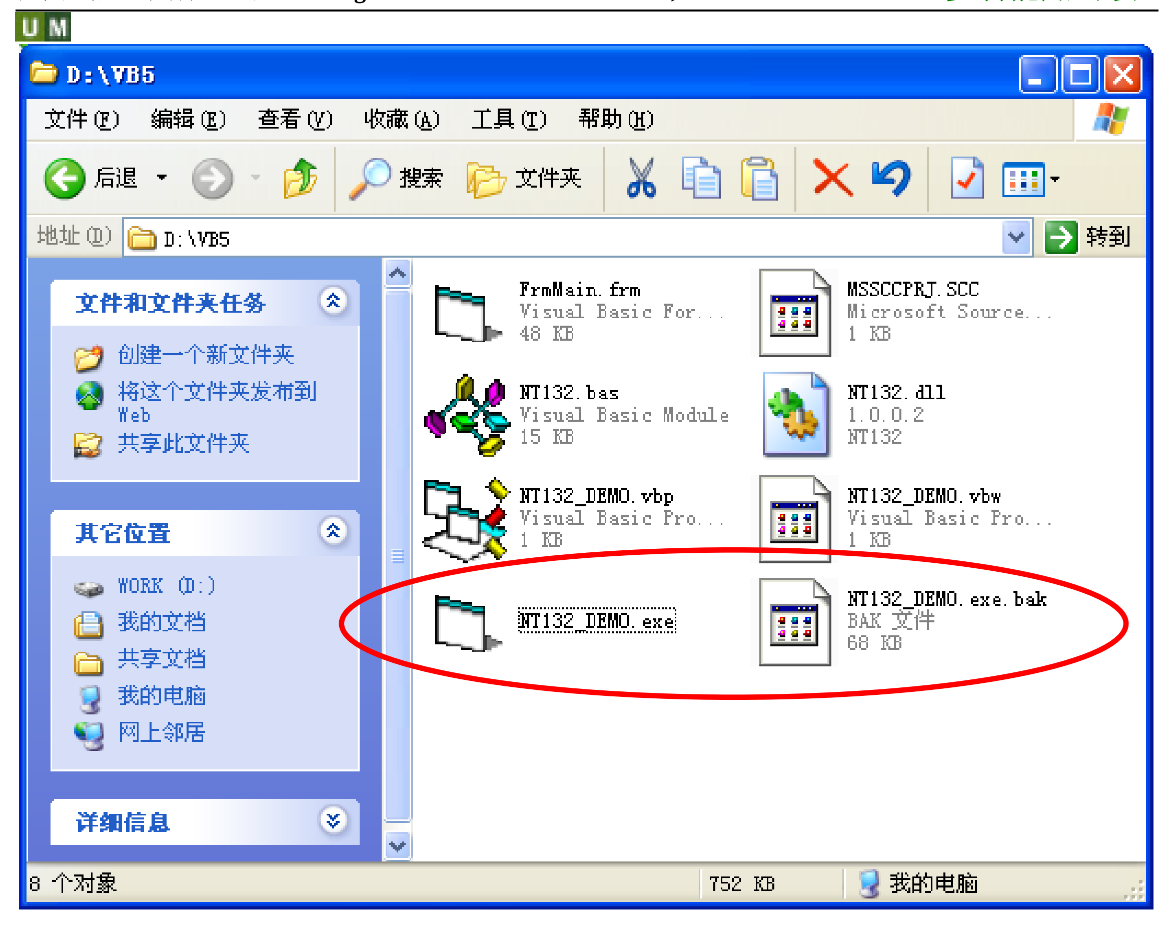


图 7.20　DLL 外壳加密结果

11. 将NT132_DEMO.exe复制到D:\下，然后双击运行它，如果插入有合法的加密锁，NT132_DEMO.exe即可正确运行，如 图 7.21 所示。

说明：经过**DLL**外壳加密后的**EXE**文件可以独立使用，所以软件打包时不需要加入**NT132.DLL**文件。

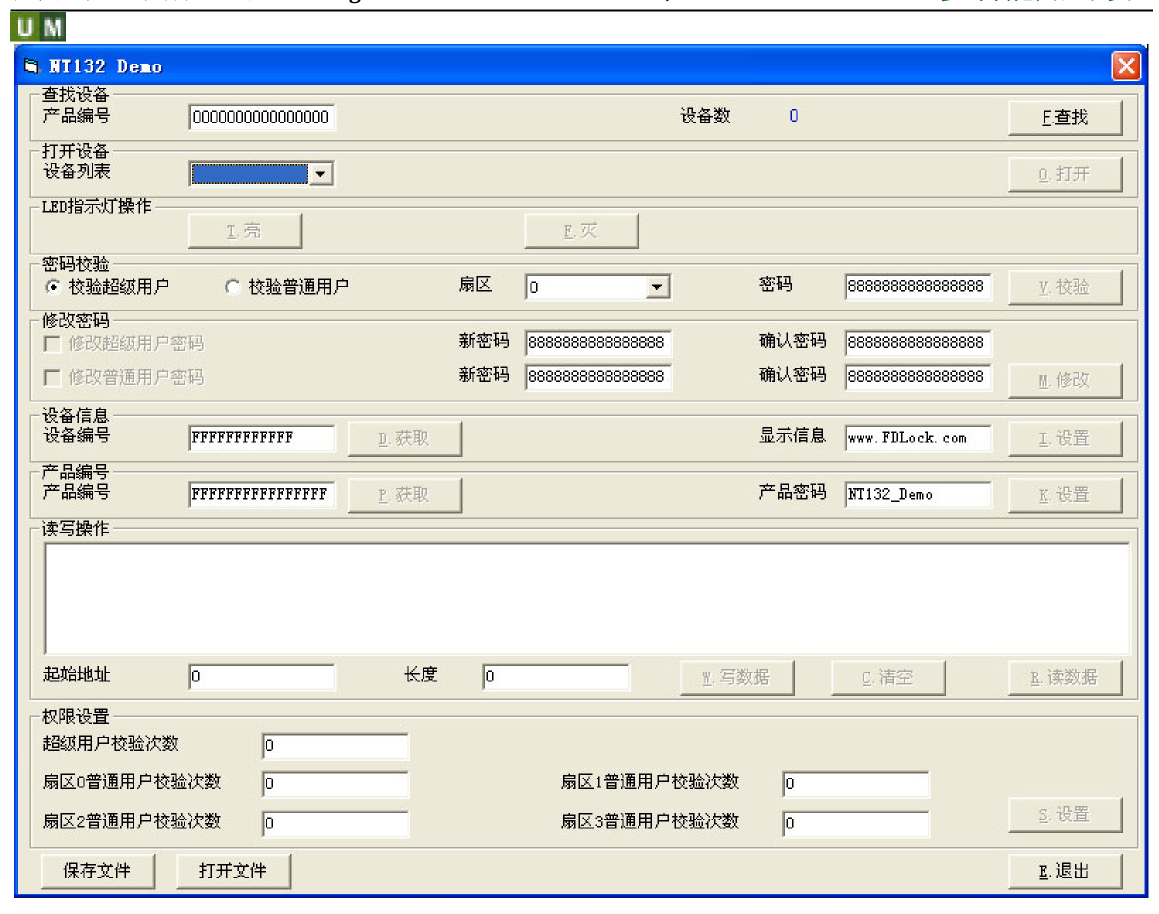


图 7.21 NT132_DEMO.exe 正确运行

8 联系我们

用户反馈

我们非常欢迎用户对我们的产品的任何反馈,您可将您对我们的产品的任何建议和意见发送 到以下 **Email** 邮箱或打电话与我们直接联系，我们会给您满意的答复。

电话：**(020)32896151/39885802-805**

Email：**Manager@ FDLock.com**

申请试用

如果您对我们的产品感兴趣，您可先申请试用，在您测试通过后再进行购买。如果您希望进行产品试用，可以到下网站填写试用单或直接打电话与我们联系：

试用单网址：**http://www. FDLock.com/Trial/index.htm**

电话：**(020) 32896151/39885802-801**

传真：**(020) 32896151/39885802-803**

Email: Trial@FDLock.com

技术支持

我们提供了多种方式的技术支持服务,您可通过如下方式与我们的技术人员咨询您所关注的技术问题：

电话：**(020) 32896151/39885802-802**

传真：**(020) 32896151/39885802-803**
Email: Support@FDLock.com

购买产品

如果您希望咨询我们产品价格或正式购买我们的产品,可以通过如下方式与我们的销售人员 联系：

电话：**(020) 32896151/39885802-801**

传真：**(020) 32896151/39885802-803**

Email: Sales@FDLock.com